This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3222533

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 enterprise network traffic on the web is encrypted by SSL/TLS. 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× 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 approaches motivate us to develop lightweight middleboxes

- *Qin Liu, Yu Peng and Hongbo Jiang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan Province, P.R. China, 410082. E-mail: gracelq628@hnu.edu.cn; pengyu411@hnu.edu.cn; hongbojiang2004@gmail.com*
- *Jie Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA. E-mail: jiewu@temple.edu*
- *Tian Wang is with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University & UIC, Zhuhai, Guangdong Province, P. R. China, 519000. E-mail: cs_tianwang@163.com*
- *Tao Peng and Guojun Wang are with the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, Guangdong Province, P.R. China, 510006. E-mail: pengtao@gzhu.edu.cn; csgjwang@gzhu.edu.cn*

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, 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, 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 SlimBox, a privacy-preserving and lightweight middlebox, which supports pattern matching over encrypted traffic while incurring moderate overheads under well-defined leakage.
- We construct SlimBox under different trade-off between security and efficiency. Compared with existing SE-based solutions, 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 SlimBox and evaluate the performance on real-world datasets. Experimental results demonstrate that 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 SlimBox^0 and 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 hardware 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, Embark [12] enhances the token matching technique to realize prefix matching; BilndIDS [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.

TABLE 1: SE-based middleboxes.

Scheme	Matching speed	Communication overhead	Fast filtering
BlindBox [11]	Fast	High	×
PrivDP [16]	Fast	High	×
SEST [18]	Slow	High	×
S4E [19]	Slow	High	×
SlimBox^0	Fast	High	✓
SlimBox*	Fast	Moderate	✓
SlimBox* (two-round)	Fast	Small	✓

Recently, another line of work employed bilinear pairings to support pattern matching against the encrypted data. SEST [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 SEST. 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 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 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.

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 detection, the IGW first sends the packets to the MB before

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3222533

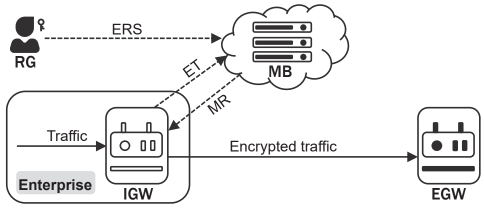


Fig. 1: System architecture. The communication channels between gateways are assumed to be secured under SSL/TLS.

interacting with the EGW. The rule/payload normally contains sensitive information like trade secrets and intellectual property, and will be encrypted before being passed through the MB. The MB that centralizes abundant resources is employed to perform pattern-matching-based inspection over the encrypted traffic (ET) and encrypted rules (ERS). Once the matching succeeds, the MB determines the packet is abnormal and returns matching results $\mathcal{MR}$ to IGW that will take further action according to $\mathcal{MR}$.

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 mismatching 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 $\mathbf{B}$, its length is denoted by $|\mathbf{B}|$ and the i-th element by $\mathbf{B}[i]$ for $i \in [|\mathbf{B}|]$. 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 $0\langle 1 \cdots 1 \rangle_x 0$. A character string $\mathcal{S}$ is defined on an universe character set Σ (e.g., ASCII character set), with $|\mathcal{S}|$ denoting the number of characters contained in $\mathcal{S}$. The concatenation of two strings $\mathcal{S}_1$ and $\mathcal{S}_2$ is denoted by $\mathcal{S}_1 || \mathcal{S}_2$. For a finite set X, its cardinality is denoted by $|X|$, and $(x_1, \ldots, x_n) \xleftarrow{\$} X$ 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 $\mathbb{G}$ be a prime order cyclic group of order p generated by g. The DDH assumption holds in $\mathbb{G}$, if $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{ddh}}(\lambda) = \Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c) = 1]$ is negligible for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ and any a, b, c randomly chosen from $\mathbb{Z}_p^*$.

Lemma 1 [22]. For any integers α, β, and any PPT adversary $\mathcal{A}$, the probability of $\Pr[\mathcal{A}(g, g^{\mathbf{a}}, g^{\mathbf{b}}, g^{\mathbf{ab}^T}) = 1] - \Pr[\mathcal{A}(g, g^{\mathbf{a}}, g^{\mathbf{b}}, \mathbf{M}) = 1]$ is negligible, where vector $\mathbf{a} \in (\mathbb{Z}_p^*)^\alpha$, vector $\mathbf{b} \in (\mathbb{Z}_p^*)^\beta$, $g^{\mathbf{ab}^T} \in \mathbb{G}^{\alpha \times \beta}$, and $\mathbf{M}$ is uniform over $\mathbb{G}^{\alpha \times \beta}$.

Symmetric Key Encryption (SKE). It consists of two polynomial-time algorithms $\mathsf{SKE} = (\mathsf{Enc}, \mathsf{Dec})$. 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 $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{ske}}(\lambda)$ denote the advantage for an adversary $\mathcal{A}$ to distinguish the ciphertexts of two equal-length plaintexts. SKE is IND-CPA secure if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{ske}}(\lambda)$ is negligible.

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

The OXT scheme. The basic idea is to integrate the following cross searching process into SE. Given a conjunction of query keywords $(w_1, w_2, \ldots, w_t)$, the server first performs *inverted searching* to find out the files containing keyword w_1, denoted by $\mathsf{DB}(w_1)$, and then for each file $f \in \mathsf{DB}(w_1)$, the server performs *forward checking* to test whether f contains all the remaining keywords $w_2, \ldots, w_t$. 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 checking, 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 $F_p : \{0,1\}^\lambda \times \{0,1\}^* \to \mathbb{Z}_p^*$ 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 $\mathcal{P}$ into a set of payload pairs $\{(\mathrm{kg}_s, \mathrm{pos}_s)\}_{s=1}^{|\mathcal{P}|-k+1}$ (line 1 of algorithm PT), where the s-th pair $(\mathrm{kg}_s, \mathrm{pos}_s)$ denotes the k-gram

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3222533

Algorithm 1 Pretreatment Phase

Preprocessing Traffic (PT)

Input: Payload string $\mathcal{P}$, length k, secret key K
Output: Payload pairs $\{(\mathrm{kg}_s, \mathrm{xp}_s)\}_{s=1}^{n}$
1: Transform $\mathcal{P}$ into $\{(\mathrm{kg}_s, \mathrm{pos}_s)\}_{s=1}^{|\mathcal{P}|-k+1}$
2: $\mathrm{xp}_0 \xleftarrow{\$} \mathbb{Z}_p^*$; $n \leftarrow |\mathcal{P}| - k + 1$
3: **for** $s \in [n]$ **do**
4: $\quad \mathrm{xp}_s \leftarrow (\mathrm{xp}_{s-1} + F_p(K, \mathrm{kg}_s)) \mod p$

Preprocessing Rules (PR)

Input: Rule string $\mathcal{R}_i$, length k, secret key K
Output: Rule patterns $R_i = (\mathrm{lkg}_i, \{(\mathrm{okg}_j, \Delta_j)\}_{j=1}^{h})$
1: Transform $\mathcal{R}_i$ into $(\mathrm{lkg}_i, \{(\mathrm{okg}_j, \mathrm{ofs}_j)\}_{j=1}^{h})$
2: $R_i \leftarrow (\mathrm{lkg}_i, \{(\mathrm{okg}_j, \mathrm{ofs}_j)\}_{j=1}^{h})$; $(\Delta_1, \ldots, \Delta_h) \leftarrow 0$
3: **for** $j \in [h]$ **do**
4: $\quad$ **for** $x = 1$ **to** $|\mathrm{ofs}_j|$ **do**
5: $\quad\quad$ **if** $\mathrm{ofs}_j > 0$ **then**
6: $\quad\quad\quad$ Obtain the k-gram okg with offset x from lkg_i
7: $\quad\quad\quad$ $\Delta_j \leftarrow (\Delta_j + F_p(K, \mathrm{okg})) \mod p$
8: $\quad\quad$ **else**
9: $\quad\quad\quad$ Obtain the k-gram okg with offset $-x+1$ from lkg_i $\quad \triangleright$ $\mathrm{okg} = \mathrm{lkg}_i$ if offset equals 0
10: $\quad\quad\quad$ $\Delta_j \leftarrow (\Delta_j - F_p(K, \mathrm{okg})) \mod p$
11: $\quad$ Replace $\mathrm{ofs}_j \in R_i$ with Δ_j

kg_s located at position pos_s of $\mathcal{P}$. A rule string $\mathcal{R}_i$ is transformed into a set of *offset k-grams* with *relative offsets* regarding a *baseline k-gram* (line 1 of algorithm PR), $(\mathrm{lkg}_i, \{(\mathrm{okg}_j, \mathrm{ofs}_j)\}_{j=1}^{h})$, where lkg_i is the baseline k-gram, and $(\mathrm{okg}_j, \mathrm{ofs}_j)$ 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 $(\mathrm{kg}_s, \mathrm{pos}_s)$, the position value pos_s is replaced by $\mathrm{xp}_s = \sum_{i=1}^{s} F_p(K, \mathrm{kg}_i) + \mathrm{xp}_0$, where xp_0 is a random value from $\mathbb{Z}_p^*$ (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 xp_0 (i.e., $\mathrm{xp}_0 = F_p(K, \mathrm{t}||\mathrm{id})$). Let okg^x denote the offset k-gram with offset x from the baseline k-gram lkg_i. As shown in the line 3-11 of algorithm PR, for the pair $(\mathrm{okg}_j, \mathrm{ofs}_j) \in R_i$, the offset value ofs_j is replaced by $\Delta_j = \sum_{x=1}^{\mathrm{ofs}_j} F_p(K, \mathrm{okg}^x) \mod p$ if $\mathrm{ofs}_j > 0$, and by $\Delta_j = -\sum_{x=0}^{\mathrm{ofs}_j+1} F_p(K, \mathrm{okg}^x) \mod p$ if $\mathrm{ofs}_j < 0$ ($\mathrm{okg}^0 = \mathrm{lkg}_i$). 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 $(\mathrm{xp}_t = \mathrm{xp}_s + \Delta_j) \Leftrightarrow (\mathrm{pos}_t = \mathrm{pos}_s + \mathrm{ofs}_j)$.

With the pretreatment step, the cross searching process can be performed as follows: For each payload pair $(\mathrm{kg}_s, \mathrm{xp}_s)$, the MB first checks whether there exists a rule $(\mathrm{lkg}_i, \{(\mathrm{okg}_j, \Delta_j)\}_{j=1}^{h})$, such that $\mathrm{lkg}_i = \mathrm{kg}_s$ (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 $\mathrm{xp}_s + \Delta_j$ 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.

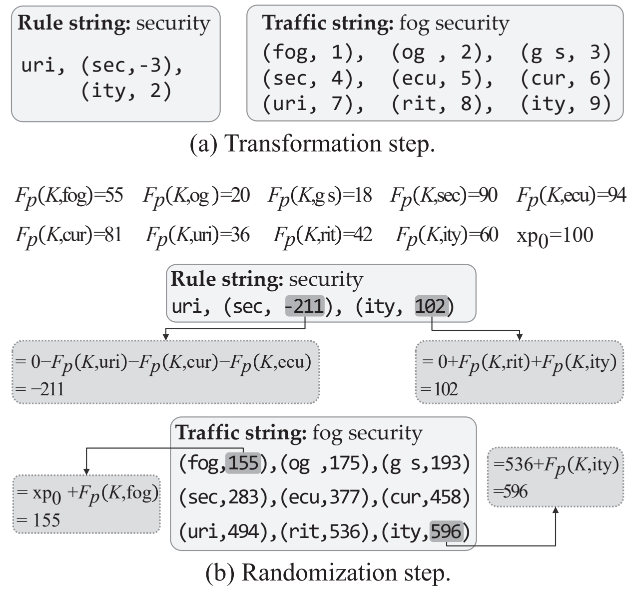


Fig. 2: A sample preprocessed rule/traffic ($k = 3$).

4.2 Syntax and Definitions

After pretreatment, the traffic is in the form of $T = (\mathrm{id}, (P_1, \ldots, P_n))$, where $\mathrm{id} \in \{0,1\}^{\lambda}$ is the traffic identifier, $n = |\mathcal{P}| - k + 1$ and the s-th payload pair $P_s = (\mathrm{kg}_s, \mathrm{xp}_s)$ for $s \in [n]$. The ruleset consists of a set of pattern/action pairs $\mathrm{RS} = \{(R_1, A_1), \ldots, (R_m, A_m)\}$, where $R_i = (\mathrm{lkg}_i, \{(\mathrm{okg}_j, \Delta_j)\}_{j=1}^{h})$[2], 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 iff $\mathrm{lkg}_i = \mathrm{kg}_s$ and $(\mathrm{okg}_j, \mathrm{xp}_s + \Delta_j)$ exists in T for $j \in [h]$. The rule R_i partially matches the s-th pair of the traffic T iff $\mathrm{lkg}_i = \mathrm{kg}_s$ and there exists $j \in [h]$ s.t. $(\mathrm{okg}_j, \mathrm{xp}_s + \Delta_j) \notin T$. The rule R_i does not match the traffic T at all iff $\mathrm{lkg}_i \neq \mathrm{kg}_s$ for all $s \in [n]$.

SlimBox consists of the following algorithms: (1) (ERS) $\leftarrow$ RuleEnc(RS, SK): The RG takes the secret keys SK and the ruleset RS as input and outputs the encrypted ruleset ERS. (2) (ET) $\leftarrow$ TrafEnc(T, LM, SK): The IGW takes the traffic T, the map LM, and the keys SK as

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.

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.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3222533

TABLE 2: Summary of Notations

k	The length of k-gram
$\mathcal{R}_i$	A rule string
lkg_i	The baseline k-gram of $\mathcal{R}_i$
$(\mathrm{okg}_j, \mathrm{ofs}_j)$	The j-th offset k-gram okg_j of a rule and its relative offset ofs_j from the baseline k-gram
$(\mathrm{okg}_j, \Delta_j)$	The j-th offset k-gram okg_j of a rule and its random offset Δ_j
ltrap_i	The label trapdoor of $\mathcal{R}_i$
$(\mathrm{xtrap}_j, \mathrm{xofs}_j)$	The j-th cross trapdoor pair of a rule
$\mathcal{P}$	A payload string
$(\mathrm{kg}_s, \mathrm{pos}_s)$	The k-gram kg_s and its position pos_s located at $\mathcal{P}$
$(\mathrm{kg}_s, \mathrm{xp}_s)$	The k-gram kg_s and its random position xp_s
$(\mathrm{ltag}_s, \mathrm{xtag}_s)$	The label tag and cross tag of k-gram kg_s
(α_s, β_s)	The auxiliary tags of k-gram kg_s

input, and outputs the encrypted traffic ET. (3) $(\mathcal{MR}) \leftarrow$ Match(ERS, ET): The MB takes the encrypted ruleset ERS and the encrypted traffic ET as input and outputs the matching result $\mathcal{MR}$. (4) $(\mathrm{AS}) \leftarrow$ Action($\mathcal{MR}, SK$): The IGW takes the result $\mathcal{MR}$ and the keys SK as input and recovers the actions AS.

Specifically, given a set of rule patterns $(\mathrm{lkg}_i, \{(\mathrm{okg}_j, \Delta_j)\}_{j=1}^h)$ of the rule string $\mathcal{R}_i$, 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 Δ_j will be encrypted into cross trapdoor pair $(\mathrm{xtrap}_j, \mathrm{xofs}_j)$. Given a payload pair $(\mathrm{kg}_s, \mathrm{xp}_s)$ of the payload string $\mathcal{P}$, the TrafEnc algorithm will encrypt kg_s into label tag ltag_s along with generating the cross tag xtag_s and the auxiliary tags (α_s, β_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 $\Delta_j + \mathrm{xp_s}$ 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 $\{0,1\}^*$ and $\mathbb{Z}_p^*$, respectively. For ease of illustration, we omit the PRF keys and consider a basic extension: For each pattern $R_i = (\mathrm{lkg}_i, \{(\mathrm{okg}_j, \Delta_j)\}_{j=1}^h)$, the RG generates a label trapdoor $\mathrm{ltrap}_i = F(\mathrm{lkg}_i)$, and a set of cross trapdoors $\{(\mathrm{xtrap}_j, \mathrm{xofs}_j)\}_{j=1}^h$, where $\mathrm{xtrap}_j = g^{F_p(\mathrm{okg}_j)\times v}$ and $\mathrm{xofs}_j = g^{F_p(\mathrm{okg}_j)\times \Delta_j \times z^{-1}}$. For each pair $(\mathrm{kg}_s, \mathrm{xp}_s) \in T$, the IGW generates a label tag $\mathrm{ltag}_s = F(\mathrm{kg}_s)$, a cross tag $\mathrm{xtag}_s = g^{F_p(\mathrm{kg})\times F_p(\mathrm{id})\times \mathrm{xp}_s}$, and two auxiliary tags $\alpha_s = F_p(\mathrm{id}) \times z$ and $\beta_s = F_p(\mathrm{id}) \times \mathrm{xp}_s \times v^{-1}$. As OXT, z and v are two blinding factors which can be cancelled during matching only when $\mathrm{kg}_s = \mathrm{lkg}_i$. If there is a match, the MB further calculates $\mathrm{xtrap}_j^{\beta_s} \times \mathrm{xofs}_j^{\alpha_s}$ to obtain $g^{F_p(\mathrm{okg}_j)\times F_p(\mathrm{id})\times(\mathrm{xp}_s+\Delta_j)}$, and tests if the computed result exists in the cross tag set.

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 $R_i = (\mathrm{lkg}_i, \mathrm{okg}_i, \Delta_i)$ and $R_j = (\mathrm{lkg}_j, \mathrm{okg}_j, \Delta_j)$ match the s-th pair and the t-th pair of the traffic T, respectively. In this case, the MB can obtain four intermediate values $V_1 = g^{F_p(\mathrm{okg}_i)\times F_p(\mathrm{id})\times \mathrm{xp}_s}$, $V_2 = g^{F_p(\mathrm{okg}_i)\times F_p(\mathrm{id})\times \Delta_i}$, $V_3 = g^{F_p(\mathrm{okg}_j)\times F_p(\mathrm{id})\times \mathrm{xp}_t}$ and $V_4 = g^{F_p(\mathrm{okg}_j)\times F_p(\mathrm{id})\times \Delta_j}$. If $\mathrm{okg}_i = \mathrm{okg}_j$, the MB can obtain the cross tags for pairs $(\mathrm{okg}_i, \mathrm{xp}_s + \Delta_j)$ and $(\mathrm{okg}_i, \mathrm{xp}_t + \Delta_i)$, by calculating $V_1 \times V_4$ and $V_2 \times V_3$, respectively. To avoid the extra leakage, the position/offset value is associated with a new blinding factor, which can be cancelled only when the forward checking is correctly performed.

5.2 Basic Construction

Let $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ and $F_p : \{0,1\}^\lambda \times \{0,1\}^* \to \mathbb{Z}_p^*$ be PRFs, and let SKE = (Enc, 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 $SK = (K_e, K_I, K_S, K_Z, K_V, K_L, K_X)$ randomly chosen from $\{0,1\}^\lambda$, 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 $\{(\mathrm{xtrap}_j, \mathrm{xofs}_j)\}_{j=1}^h$, and an encrypted action $\mathbf{cp}_i$ are generated, where ltrap_i and the blinding factors hiding cross trapdoors are calculated based on $\mathrm{lkg}_i || c$. In algorithm TrafEnc^0, the s-th pair $(\mathrm{kg}_s, \mathrm{xp}_s)$ is encrypted into a label tag ltag_s, a cross tag xtag_s, and two auxiliary tags (α_s, β_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 $(\mathrm{id}, \mathrm{kg}_s, \mathrm{xp}_s)$, 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 $\mathsf{ERS}[\mathrm{ltag}_s] = \perp$ or not, for $(\mathrm{ltag}_s, \alpha_s, \beta_s)$ in the encrypted traffic. If so, this means that there is no rule that matches the s-th pair $(\mathrm{kg}_s, \mathrm{xp}_s)$ of the traffic and the MB continues to check the next pair. Otherwise, this means that there exists a rule $R_i = (\mathrm{lkg}_i, \{(\mathrm{okg}_j, \Delta_j)\}_{j=1}^h)$ such that $\mathrm{lkg}_i = \mathrm{kg}_s$. Then, the MB further calculates the cross tags $\mathrm{xtrap}_j^{\beta_s} \times \mathrm{xofs}_j^{\alpha_s}$ for k-gram/position pairs $\{(\mathrm{okg}_j, \mathrm{xp}_s + \Delta_j)\}_{j=1}^h$. Note that when $\mathsf{ERS}[\mathrm{ltag}_s] \neq \perp$, the blinding factors z, v, l used in auxiliary tags and cross trapdoors are the same. Therefore, we have:

$$
\begin{aligned}
&\mathrm{xtrap}_j^{\alpha_s} \times \mathrm{xofs}_j^{\beta_s} \\
&= g^{\mathrm{xokg}_j \times v \times \mathrm{xid} \times (\mathrm{xp}_s + l) \times v^{-1}} \times g^{\mathrm{xokg}_j \times (\Delta_j - l) \times z^{-1} \times \mathrm{xid} \times z} \\
&= g^{\mathrm{xokg}_j \times \mathrm{xid} \times (\mathrm{xp}_s + l)} \times g^{\mathrm{xokg}_j \times \mathrm{xid} \times (\Delta_j - l)} \\
&= g^{\mathrm{xokg}_j \times \mathrm{xid} \times (\mathrm{xp}_s + \Delta_j)},
\end{aligned} \tag{1}
$$

where $\mathrm{xokg}_j = F_p(K_X, \mathrm{okg}_j)$.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3222533

Algorithm 2 Basic SlimBox Construction

<u>RuleEnc^0</u>

Input: Ruleset RS, secret keys SK
Output: Encrypted ruleset ERS

Parse SK as $(K_e, K_I, K_S, K_Z, K_V, K_L, K_X)$
$\mathsf{ERS} \leftarrow$ empty map
for each $(R_i, A_i) \in \mathrm{RS}$ **do**
  Parse R_i as $(\mathrm{lkg}_i, (\mathrm{okg}_1, \Delta_1), \ldots, (\mathrm{okg}_h, \Delta_h))$
  for $c = 1$ to MAX **do**
    $\mathrm{ltrap}_i \leftarrow F(K_S, \mathrm{lkg}_i \| c)$
    $\mathbf{cp}_i \leftarrow \mathsf{SKE.Enc}(K_e, A_i)$; $z \leftarrow F_p(K_Z, \mathrm{lkg}_i \| c)$
    $v \leftarrow F_p(K_V, \mathrm{lkg}_i \| c)$; $l \leftarrow F_p(K_L, \mathrm{lkg}_i \| c)$
    for $j = 1$ to h **do**
      $\mathrm{xtrap}_j \leftarrow g^{F_p(K_X, \mathrm{okg}_j) \times v}$
      $\mathrm{xofs}_j \leftarrow g^{F_p(K_X, \mathrm{okg}_j) \times (\Delta_j - l) \times z^{-1}}$
    $\mathsf{ERS}[\mathrm{ltrap}_i] \leftarrow (\{(\mathrm{xtrap}_j, \mathrm{xofs}_j)\}_{j=1}^h, \mathbf{cp}_i)$

<u>TrafEnc^0</u>

Input: Traffic T, local map LM, secret keys SK
Output: Encrypted traffic ET

Parse SK as $(K_e, K_I, K_S, K_Z, K_V, K_L, K_X)$
Parse T as $(\mathrm{id}, \{(\mathrm{kg}_s, \mathrm{xp}_s)\}_{s=1}^n)$
$(\mathrm{LT}, \mathrm{XT}, \mathsf{ET}) \leftarrow \emptyset$; $\mathrm{xid} \leftarrow F_p(K_I, \mathrm{id})$
for the s-th pair $(\mathrm{kg}_s, \mathrm{xp}_s) \in T$ **do**
  if $\mathsf{LM}[\mathrm{kg}_s] = \perp$ **then**
    $c \leftarrow 1$
  else
    $c \leftarrow \mathsf{LM}[\mathrm{kg}_s]$
  $\mathsf{LM}[\mathrm{kg}_s] \leftarrow c + 1$
  $\mathrm{xtag}_s \leftarrow g^{F_p(K_X, \mathrm{kg}_s) \times \mathrm{xp}_s \times \mathrm{xid}}$; $z \leftarrow F_p(K_Z, \mathrm{kg}_s \| c)$
  $v \leftarrow F_p(K_V, \mathrm{kg}_s \| c)$; $l \leftarrow F_p(K_L, \mathrm{kg}_s \| c)$
  $\alpha_s \leftarrow \mathrm{xid} \times z$; $\beta_s \leftarrow \mathrm{xid} \times (\mathrm{xp}_s + l) \times v^{-1}$
  $\mathrm{ltag}_s \leftarrow F(K_S, \mathrm{kg}_s \| c)$
  $\mathrm{LT} \leftarrow \mathrm{LT} \cup (\mathrm{ltag}_s, \alpha_s, \beta_s)$; $\mathrm{XT} \leftarrow \mathrm{XT} \cup \mathrm{xtag}_s$
$\mathsf{ET} \leftarrow (\mathrm{LT}, \mathrm{XT})$

<u>Match^0</u>

Input: Encrypted ruleset ERS, encrypted traffic ET
Output: Matching results $\mathcal{MR}$

$\mathcal{MR} \leftarrow \emptyset$; parse ET as $(\mathrm{LT}, \mathrm{XT})$
for each $(\mathrm{ltag}, \alpha, \beta) \in \mathrm{LT}$ **do**
  if $\mathsf{ERS}[\mathrm{ltag}] \neq \perp$ **then**
    obtain $(\{(\mathrm{xtrap}_j, \mathrm{xofs}_j)\}_{j=1}^h, \mathbf{cp})$ from $\mathsf{ERS}[\mathrm{ltag}]$
    for $j = 1$ to h **do**
      $x_j \leftarrow \mathrm{xtrap}_j^\beta \times \mathrm{xofs}_j^\alpha$
    if $x_1, \ldots, x_h \in \mathrm{XT}$ **then**
      $\mathcal{MR} \leftarrow \mathcal{MR} \cup (\mathrm{ltag}, \mathbf{cp})$

<u>Action^0</u>

The IGW runs $\mathsf{SKE.Dec}$ to recover actions in $\mathcal{MR}$

From Eq. (1), the MB can check whether $(\mathrm{okg}_j, \mathrm{xp}_s + \Delta_j)$ 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 $R_i = (\mathrm{lkg}_i, \{(\mathrm{okg}_j, \Delta_j)\}_{j=1}^h)$, algorithm RuleEnc^0 requires $2h \times \mathsf{MAX}$ exponentiations in $\mathbb{G}$ for generating cross trapdoors. For a traffic $T = (\mathrm{id}, (P_1, \ldots, P_n))$, algorithm TrafEnc^0 requires n exponentiations in $\mathbb{G}$ for generating cross tags. If R_i fully/partially matches the s-pair of traffic T, algorithm Match^0 requires $2h$ exponentiations and h multiplication in $\mathbb{G}$. Otherwise, no group-related operation is required. As for communication costs, the encrypted ruleset contains $m \times \mathsf{MAX}$ entries, where each entry contains two λ-bit strings and $2h$ elements in $\mathbb{G}$. The encrypted traffic consists of two sets LT and XT, where LT contains n λ-bit strings and $2n$ elements in $\mathbb{Z}_p^*$, and XT contains n elements in $\mathbb{G}$. The matching results $\mathcal{MR}$ contains two λ-bit strings.

6 The Design of SlimBox*

6.1 General Ideas

SlimBox^0 allows the MB to securely inspect traffic, but still has the following insufficiency: (1) The size of the encrypted ruleset is related to 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/mismatches 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 MAX. Therefore, there is no need for the IGW 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 generate s for each (kg, 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 (lkg, 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

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3222533

Algorithm 3 Advanced SlimBox Construction

<u>RuleEnc*</u>

Input: Ruleset RS, secret keys SK
Output: Encrypted ruleset ERS

Parse SK as $(K_e, K_I, K_S, K_Z, K_V, K_L, K_X)$
$\mathsf{ERS} \leftarrow$ empty map
for each $(R_i, A_i) \in \mathrm{RS}$ **do**
  Parse R_i as $(\mathrm{lkg}_i, (\mathrm{okg}_1, \Delta_1), \ldots, (\mathrm{okg}_h, \Delta_h))$
  Run Alg. 4 to generate a bit string $\mathbf{B}_i$
  for $c = 1$ to MAX^* **do**
    Execute lines 7-11 in the basic RuleEnc algorithm and obtain $(\{(\mathrm{xtrap}_j, \mathrm{xofs}_j)\}_{j=1}^{h}, \mathbf{cp}_i)$
    $\mathbf{cb}_i \leftarrow \mathsf{SKE.Enc}(K_e, \mathbf{B}_i)$; $\mathbf{s}_i \leftarrow F(K_S, \mathrm{lkg}_i||c||0)$
    $\mathrm{lt}_i \leftarrow F(K_S, \mathrm{lkg}_i||c||1)$; $\mathrm{ltrap}_i \leftarrow H(\mathbf{s}_i||\mathrm{lt}_i)$
    $\mathsf{ERS}[\mathrm{ltrap}_i] \leftarrow (\{(\mathrm{xtrap}_j, \mathrm{xofs}_j)\}_{j=1}^{h}, \mathrm{lt}_i, \mathbf{s}_i, \mathbf{cp}_i, \mathbf{cb}_i)$

<u>TrafEnc*</u>

Input: Traffic T, local map LM, secret keys SK
Output: Encrypted traffic ET, cross tag map XT

Parse SK as $(K_e, K_I, K_S, K_Z, K_V, K_L, K_X)$
Parse T as $(\mathrm{id}, \{(\mathrm{kg}_s, \mathrm{xp}_s)\}_{s=1}^{n})$
$\mathsf{ET} \leftarrow \emptyset$; $(\mathsf{XT}, \mathsf{EM}) \leftarrow$ empty map; $\mathrm{xid} \leftarrow F_p(K_I, \mathrm{id})$
for the s-th pair $(\mathrm{kg}_s, \mathrm{xp}_s) \in T$ **do**
  if $\mathsf{EM}[\mathrm{kg}_s] = \perp$ **then**
    $c \leftarrow 1$
  else
    $c \leftarrow \mathsf{EM}[\mathrm{kg}_s]$
  $\mathsf{EM}[\mathrm{kg}_s] \leftarrow c + 1$
  Execute lines 10-12 to obtain $\mathrm{xtag}_s, \alpha_s, \beta_s$
  if $\mathsf{LM}[\mathrm{kg}_s||c] = \perp$ **then**
    $\mathbf{s} \leftarrow F(K_S, \mathrm{kg}_s||c||0)$
  else
    $\mathbf{s} \leftarrow \mathsf{LM}[\mathrm{kg}_s||c]$
  $\mathsf{LM}[\mathrm{kg}_s||c] \leftarrow \mathbf{s} + 1$; $\mathrm{ltag}_s \leftarrow H(\mathbf{s}||F(K_S, \mathrm{kg}_s||c||1))$
  $\mathsf{ET} \leftarrow \mathsf{ET} \cup (\mathrm{ltag}_s, \alpha_s, \beta_s)$; $\mathsf{XT}[s] \leftarrow (\mathrm{ltag}_s, \mathrm{xtag}_s)$

<u>Match*</u>

Input: Encrypted ruleset ERS, encrypted traffic ET
Output: Matching results $\mathcal{MR}$

$\mathcal{MR} \leftarrow \emptyset$
for each $(\mathrm{ltag}, \alpha, \beta) \in \mathsf{ET}$ **do**
  if $\mathsf{ERS}[\mathrm{ltag}] \neq \perp$ **then**
    $((\mathrm{xtrap}_j, \mathrm{xofs}_j)_{j=1}^{h}, \mathrm{lt}, \mathbf{s}, \mathbf{cp}, \mathbf{cb}) \leftarrow \mathsf{ERS}[\mathrm{ltag}]$
    $\mathsf{ERS}[\mathrm{ltag}] \leftarrow \perp$; $\mathbf{s} \leftarrow \mathbf{s} + 1$; $\mathrm{ltrap} \leftarrow H(\mathbf{s}||\mathrm{lt})$
    $\mathsf{ERS}[\mathrm{ltrap}] \leftarrow ((\mathrm{xtrap}_j, \mathrm{xofs}_j)_{j=1}^{h}, \mathrm{lt}, \mathbf{s}, \mathbf{cp}, \mathbf{cb})$
    $\widetilde{\mathbf{cp}} \leftarrow \mathbf{cp} \oplus_{j=1}^{h} H(\mathrm{xtrap}_j^{\beta} \times \mathrm{xofs}_j^{\alpha})$
    $\mathcal{MR} \leftarrow \mathcal{MR} \cup (\mathrm{ltag}, \widetilde{\mathbf{cp}}, \mathbf{cb})$

<u>Action*</u>

Input: Matching results $\mathcal{MR}$, secret keys SK
Output: Action set AS

$\mathrm{AS} \leftarrow \emptyset$
for each $(\mathrm{ltag}, \widetilde{\mathbf{cp}}, \mathbf{cb}) \in \mathcal{MR}$ **do**
  Locates $\mathsf{XT}[s]$ s.t $\mathrm{ltag} \in \mathsf{XT}[s]$
  Run $\mathsf{SKE.Dec}(K_e, \mathbf{cb})$ to recover the bit string $\mathbf{B}$
  for each offset o obtained from $\mathbf{B}$ **do**
    Obtain xtag from $\mathsf{XT}[s + o]$; $\widetilde{\mathbf{cp}} \leftarrow \widetilde{\mathbf{cp}} \oplus \mathrm{xtag}$
  Run $\mathsf{SKE.Dec}(K_e, \widetilde{\mathbf{cp}})$ to recover the action A
  $\mathrm{AS} \leftarrow \mathrm{AS} \cup A$

auxiliary tags needs to be transmitted, the traffic size is greatly decreased.

6.2 Advanced Construction

Let $F : \{0,1\}^\lambda \times \{0,1\}^* \rightarrow \{0,1\}^\lambda$ and $F_p : \{0,1\}^\lambda \times \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be PRFs, and let $\mathsf{SKE} = (\mathsf{Enc}, \mathsf{Dec})$ be a SKE scheme. Let $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ be a collision-free hash function, and let 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/counter pair. Given $SK = (K_e, K_I, K_S, K_Z, K_V, K_L, K_X)$ defined as the basic construction, the detailed construction of $\mathsf{SlimBox}^*$ 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 $(R_i, A_i) \in \mathrm{RS}$, algorithm $\mathsf{RuleEnc}^*$ first runs Alg. 4 to construct a bit string $\mathbf{B}_i$, and then encrypts $(R_i, A_i, \mathbf{B}_i)$ for MAX^* times. In the c-th encryption, a label trapdoor ltrap_i, h cross trapdoor pairs $\{(\mathrm{xtrap}_j, \mathrm{xofs}_j)\}_{j=1}^{h}$, an auxiliary trapdoor lt_i, a random salt $\mathbf{s}_i$, an encrypted action $\mathbf{cp}_i$, and an encrypted bit string $\mathbf{cb}_i$ are generated. The main difference from algorithm $\mathsf{RuleEnc}^0$ is that the label trapdoor is computed by $\mathrm{ltrap}_i \leftarrow H(\mathbf{s}_i||\mathrm{lt}_i)$, where the random salt $\mathbf{s}_i$ is initialized by $F(K_S, \mathrm{lkg}_i||c||0)$ and will be incremented by 1 once a packet matches the rule, and $\mathrm{lt}_i = F(K_S, \mathrm{lkg}_i||c||1)$.

Algorithm 4 Encoding Bit String

Input: Rule string $\mathcal{R}$, security parameter λ, preprocessed rule pattern $(\mathrm{lkg}, \{(\mathrm{okg}_j, \Delta_j)\}_{j=1}^{h})$
Output: Bit string $\mathbf{B}$ of length λ

Initialize $\mathbf{B}$ with λ zeros
Find the position θ of lkg in the rule string $\mathcal{R}$
Set the θ-th, …, $(\theta + h + 2)$-th bits of $\mathbf{B}$ with pattern $0||\langle 1 \cdots 1 \rangle_{h+1}||0$
for $j \in [h]$ **do**
  obtain the offset o_j of okg_j from lkg in $\mathcal{R}$
  if $o_j < 0$ **then**
    Set the $(\theta + o_j)$-th bit of $\mathbf{B}$ with 1
  else
    Set the $(\theta + h + 2 + o_j)$-th bit of $\mathbf{B}$ with 1

Like algorithm $\mathsf{TrafEnc}^0$, the $\mathsf{TrafEnc}^*$ algorithm also encrypts the s-th pair $(\mathrm{kg}_s, \mathrm{xp}_s)$ into a label tag ltag_s, a cross tag xtag_s, and two auxiliary tags (α_s, β_s). The main difference is that the label tag is computed by $\mathrm{ltag}_s \leftarrow H(\mathbf{s}||F(K_S, \mathrm{kg}_s||c||1)$, where $\mathbf{s}$ is a random salt initialized by $F(K_S, \mathrm{kg}_s||c||0)$ and will be incremented by 1 once the (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, $\mathsf{XT}[s]$ keeps tags $(\mathrm{ltag}_s, \mathrm{xtag}_s)$ for the s-th pair. Thus, if a rule $R_i = (\mathrm{lkg}_i, \{(\mathrm{okg}_j, \Delta_j)\}_{j=1}^{h})$ fully matches the s-th pair, then for each offset k-gram okg_j with offset

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3222533

o from the baseline k-gram lkg_i, the calculated cross tag $\mathrm{xtrap}_j^{\beta_s} \times \mathrm{xofs}_j^{\alpha_s}$ should be equal to that stored at $\mathsf{XS}[s+o]$.

The Match^* 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 $\mathsf{RuleEnc}^*$ and $\mathsf{TrafEnc}^*$, the inverted searching can be performed correctly. However, unlike algorithm Match^0 directly returning the action ciphertexts of fully matched rules, Match^* returns the encrypted bit string $\mathbf{cb}$ and the processed ciphertext $\widetilde{\mathbf{cp}} = \mathbf{cp} \oplus_{j=1}^{h} H(\mathrm{xtrap}_j^{\beta} \times \mathrm{xofs}_j^{\alpha})$ for each rule passing inverted searching. In forward checking, the IGW first decrypts $\mathbf{cb}$ to obtain offset information and then recovers $\mathbf{cp}$ if all the calculated cross tags can be found in the map 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 XT based on the bit string.

Now, we will analyze the correctness of the encoding algorithm. Assume that $\lambda \geq h + \max|\mathcal{R}| + 3$, and that the baseline k-gram is located at position θ of the rule string. Intuitively, the pattern $0\langle 1\cdots 1\rangle_{h+1}0$ 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 $0\langle 1\cdots 1\rangle_{h+1}0$ and obtain their positions $\theta, \ldots, (\theta+h+2)$ in $\mathbf{B}$. For $i < \theta$, $\mathbf{B}[i] = 1$ denotes the offset from the label k-gram is $-(\theta - i)$, and for $i > \theta+h+2$, $\mathbf{B}[i] = 1$ denotes the offset is $i-\theta-h-2$. For example, for the rule string "security" and pattern ("uri", ("sec", -3), ("ity", 2)), the first ten bits of $\mathbf{B}$ 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 $\mathbb{G}$. The advanced algorithms consume the same complexity as the basic algorithms. As for communication costs, the encrypted ruleset contains $m \times \mathsf{MAX}^*$ entries, where each entry contains four λ-bit strings and $2h$ elements in $\mathbb{G}$. The encrypted traffic only consists of a set ET, which contains n λ-bit strings and $2n$ elements in $\mathbb{Z}_p^*$, 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 $\mathcal{MR}$ contains three λ-bit strings. The latency of traffic processing is sensitive to bandwidth. Since the size of encrypted packets is significantly declined, $\mathsf{SlimBox}^*$ offers better user experience than $\mathsf{SlimBox}^0$.

7 Security Analysis

We will analyze the security of $\mathsf{SlimBox}$ from the aspects of traffic and rule confidentiality. Since $\mathsf{SlimBox}^*$ with enhanced privacy is constructed based on $\mathsf{SlimBox}^0$, we will focus on the security of $\mathsf{SlimBox}^0$. To simplify presentation, we consider a simple scenario, where each rule R_i consists of two k-grams only, and is expressed as $(\mathrm{lkg}_i, \mathrm{okg}_i, \Delta_i)$. 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 ruleset size will not be expanded by MAX 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 $\mathcal{A}_1$ be an adversary attempting to learn the sensitive data in the traffic, and let $\mathcal{S}_1$ be a simulator parameterized by a leakage function $\mathcal{L}_1 = (n, \mathsf{MR}, \mathsf{IR})$ defined as below:

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

$\mathsf{SlimBox}^0$ achieves traffic confidentiality if $\mathcal{A}_1$ cannot distinguish the following experiments:

- $\mathbf{Real}_{\mathcal{A}_1}(\lambda)$: On input the traffic T, the experiment runs algorithm $\mathsf{TrafEnc}$ and gives ET to $\mathcal{A}_1$. For a ruleset RS adaptively chosen by $\mathcal{A}_1$, the experiment runs algorithm $\mathsf{RuleEnc}$ and gives ERS to $\mathcal{A}_1$. Finally, $\mathcal{A}_1$ outputs a bit $b \in \{0,1\}$.
- $\mathbf{Ideal}_{\mathcal{A}_1}^{\mathcal{S}_1}(\lambda)$: On input the traffic T, $\mathcal{S}_1$ generates ET with the given leakage $\mathcal{L}_1$ for $\mathcal{A}_1$. For a ruleset RS adaptively chosen by $\mathcal{A}_1$, $\mathcal{S}$ generates ERS with leakage $\mathcal{L}_1$. Finally, $\mathcal{A}_1$ 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 $\mathbb{G}$, then $\mathsf{SlimBox}^0$ is $\mathcal{L}_1$-secure against adversary $\mathcal{A}_1$.

Proof sketch. Given the leakage $\mathcal{L}_1$, the simulator $\mathcal{S}_1$ first calculates the *partition index* for each rule. Let $\hat{i}$ denote the partition index of rule R_i. First, it computes a relation $\equiv$ on rules by defining $\hat{i} \equiv \hat{j}$ iff $\mathsf{IR}[i,j] \neq 0$, 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 $\mathcal{A}_1$, $\mathcal{S}_1$ computes tags by setting $\mathrm{ltag}_s \xleftarrow{\$} \{0,1\}^\lambda$, $\mathrm{xtag}_s \xleftarrow{\$} \mathbb{G}$, and $(\alpha_s, \beta_s) \xleftarrow{\$} \mathbb{Z}_p^*$ for $s \in [n]$. To simulate ERS for an adaptively chosen ruleset RS, $\mathcal{S}_1$ needs to maintain a map X to record the calculated cross tag for each partition. For rule $R_i \in$ RS, $\mathcal{S}_1$ generates trapdoors according to the following cases: (1) $\mathsf{MR}[i] = 0$, $\mathcal{S}_1$ generates trapdoors by setting $\mathrm{ltrap}_i \xleftarrow{\$} \{0,1\}^\lambda$, and $(\mathrm{xtrap}_i, \mathrm{xofs}_i) \xleftarrow{\$} \mathbb{G}$. (2) $\mathsf{MR}[i] = (1,s) \vee \mathsf{MR}[i] = (2,s)$, $\mathcal{S}_1$ first checks whether $\mathsf{X}[\hat{i}] = \perp$ or not. If so, for the case of $\mathsf{MR}[i] = (1,s)$, $\mathcal{S}_1$ sets $\mathsf{X}[\hat{i}] \leftarrow \mathrm{xtag}_s$, and for the case of $\mathsf{MR}[i] = (2,s)$, $\mathcal{S}_1$ sets $\mathsf{X}[\hat{i}] \xleftarrow{\$} \mathbb{G}$. Then, $\mathcal{S}_1$ chooses two random elements $r_1, r_2 \in \mathbb{G}$, s.t. $r_1 \times r_2 = \mathsf{X}[\hat{i}]$, and calculates the trapdoors by setting $\mathrm{ltrap}_i \leftarrow \mathrm{ltag}_s$, $\mathrm{xtrap}_i \leftarrow r_1^{\beta_s^{-1}}$ and $\mathrm{xofs}_i \leftarrow r_2^{\alpha_s^{-1}}$. In both cases, the action ciphertext is set to $\mathbf{cp}_i \xleftarrow{\$} \{0,1\}^\lambda$.

Due to the security of PRFs and SKE, the simulated trapdoors and tags are indistinguishable from the real game when the DDH assumption holds in group $\mathbb{G}$. Thus, we conclude that for every adversary $\mathcal{A}_1$, it has a negligible

TABLE 3: The rule encryption time(s) under different rulesets and parameters.

MAX	SlimBox^0								SlimBox*							
	Snort				ETOpen				Snort				ETOpen			
	k=4	k=5	k=6	k=7	k=4	k=5	k=6	k=7	k=4	k=5	k=6	k=7	k=4	k=5	k=6	k=7
6	510	402	336	281	6033	4661	3879	3183	516	416	339	282	6045	4755	3911	3195
7	607	474	389	329	6973	5416	4468	3797	611	477	394	331	7063	5600	4500	3807
8	679	535	447	372	7955	6276	5217	4367	688	540	450	378	8035	6328	5256	4388
9	774	608	500	417	8733	6895	5677	4918	794	617	508	420	8950	6994	5819	4919
10	855	672	563	466	9923	7606	6302	5460	866	683	566	467	10011	7770	6443	5404

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

MAX	SlimBox^0								SlimBox*							
	Snort				ETOpen				Snort				ETOpen			
	k=4	k=5	k=6	k=7	k=4	k=5	k=6	k=7	k=4	k=5	k=6	k=7	k=4	k=5	k=6	k=7
6	9.94	7.93	6.61	5.60	114.32	91.45	76.19	64.36	11.09	9.08	7.76	6.75	125.99	103.23	87.99	76.18
7	11.60	9.25	7.71	6.53	133.37	106.69	88.88	75.09	12.94	10.59	9.05	7.88	146.98	120.43	102.66	88.88
8	13.26	10.57	8.81	7.47	152.42	121.93	101.58	85.82	14.79	12.11	10.34	9.00	167.98	137.63	117.32	101.57
9	14.91	11.90	9.91	8.40	171.48	137.17	114.28	96.55	16.64	13.62	11.63	10.13	188.98	154.84	131.99	114.27
10	16.57	13.22	11.01	9.34	190.53	152.41	126.98	107.27	18.49	15.13	12.93	11.25	209.98	172.04	146.66	126.97

probability to learn more information from the traffic than the defined leakage function $\mathcal{L}_1$. The formal proof of Theorem 1 could be found in Appendix A.

7.2 Rule Confidentiality

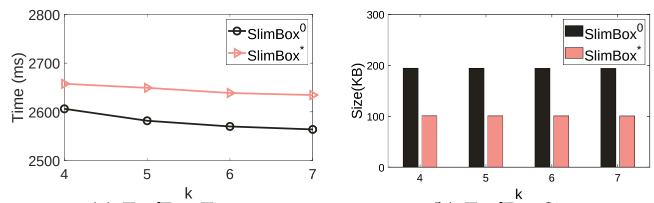
Let $\mathcal{A}_2$ be an adversary attempting to learn the sensitive data from the ruleset RS, and let $\mathcal{S}_2$ be a simulator parameterized by a leakage function $\mathcal{L}_2 = (m, n, \mathsf{MR}, \mathsf{IR})$, where (n, IR) are defined similarly as leakage function $\mathcal{L}_1$, m is the number of rules in the ruleset, and MR is the matching result of the s-th pair P_s: (1) $\mathsf{MR}[s] = 0$ denotes P_s does not match any rule at all. (2) $\mathsf{MR}[s] = (1, i)$ denotes P_s fully matches R_i. (3) $\mathsf{MR}[s] = (2, i)$ denotes P_s partially matches R_i.

SlimBox^0 achieves rule confidentiality if $\mathcal{A}_2$ cannot distinguish the following experiments:

- $\mathbf{Real}_{\mathcal{A}_2}(\lambda)$: On input the ruleset RS, the experiment runs algorithm RuleEnc and gives ERS to $\mathcal{A}_2$. For the packet T adaptively chosen by $\mathcal{A}_2$, the experiment runs algorithm TrafEnc and gives ET to $\mathcal{A}_2$. Finally, $\mathcal{A}_2$ outputs a bit $b \in \{0, 1\}$.
- $\mathbf{Ideal}^{\mathcal{S}_2}_{\mathcal{A}_2}(\lambda)$: On input the ruleset RS, $\mathcal{S}_2$ generates ERS with the given leakage $\mathcal{L}_2$ for $\mathcal{A}_2$. For the packet T adaptively chosen by $\mathcal{A}_2$, $\mathcal{S}$ generates ET with leakage $\mathcal{L}_2$. Finally, $\mathcal{A}_1$ outputs a bit $b \in \{0, 1\}$.

Theorem 2. If F, F_p are secure PRFs, SKE is CPA secure, and the DDH assumption holds in group $\mathbb{G}$, then SlimBox^0 is $\mathcal{L}_2$-secure against adversary $\mathcal{A}_2$.

Proof sketch. As the simulator $\mathcal{S}_1$, $\mathcal{S}_2$ first calculates the partition index $\hat{i}$ for each rule i based on the leakage IR. Let $\mathbf{p}$ denote the set of partition indexes. $\mathcal{S}_2$ 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 $j \in \mathbf{p}$, $\mathcal{S}_2$ sets $\mathsf{X}[j] \xleftarrow{\$} \mathbb{Z}_p^*$, and $\mathsf{F}[j] \leftarrow 0$. On input a rule R_i provided by $\mathcal{A}_2$, $\mathcal{S}_2$ generates the trapdoors by setting $\mathrm{ltrap}_i \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathrm{xtrap}_i \leftarrow g^{x_i}$ and $\mathrm{xofs}_i \leftarrow g^{y_i}$, and sets the action ciphertext as $\mathbf{cp}_i \xleftarrow{\$} \{0,1\}^\lambda$, where x_i, y_i are randomly chosen from $\mathbb{Z}_p^*$. To simulate ET for the adaptively chosen traffic T, $\mathcal{S}_2$ generates tags for the s-th payload pair P_s according to the following cases: (1) $\mathsf{MR}[s] = 0$, $\mathcal{S}_2$ sets $\mathrm{ltag}_s \xleftarrow{\$} \{0,1\}^\lambda$, $(\alpha_s, \beta_s) \xleftarrow{\$} \mathbb{Z}_p^*$, and $\mathrm{xtag}_s \xleftarrow{\$} \mathbb{G}$. (2) $\mathsf{MR}[s] = (1, i) \vee \mathsf{MR}[s] = (2, i)$, $\mathcal{S}_2$ sets the label tag as $\mathrm{ltag}_s \leftarrow \mathrm{ltrap}_i$ and sets the auxiliary tags as $\alpha_s \leftarrow r_1 \times y_i^{-1}$ and $\beta_s \leftarrow r_2 \times x_i^{-1}$, where r_1, r_2 are random elements from $\mathbb{Z}_p^*$, s.t. $r_1 + r_2 = \mathsf{X}[\hat{i}]$. If $\mathsf{MR}[s] = (1, i) \wedge \mathsf{F}[\hat{i}] = 0$, $\mathcal{S}_2$ sets the cross tag as $\mathrm{xtag}_s \leftarrow g^{\mathsf{X}[\hat{i}]}$, and updates the map as $\mathsf{F}[\hat{i}] \leftarrow 1$; Otherwise, $\mathcal{S}_2$ sets $\mathrm{xtag}_s \xleftarrow{\$} \mathbb{G}$.



(a) TrafEnc Time (b) TrafEnc Size

Fig. 3: The results of SlimBox^0 and SlimBox* for encrypting a payload with 1500 bytes with varied k.

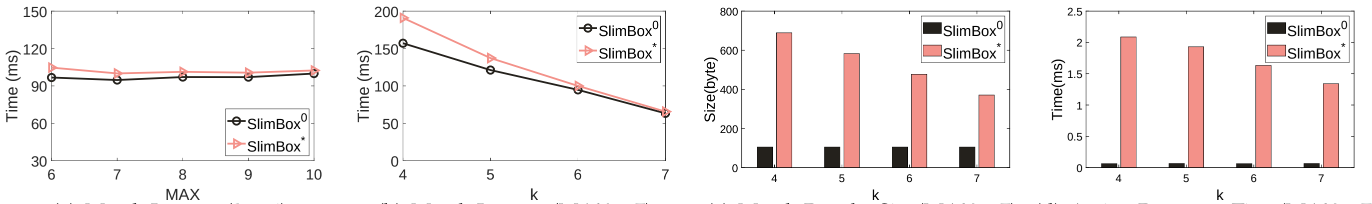
Due to the security of PRFs and SKE, the simulated trapdoors and tags are indistinguishable from the real game when the DDH assumption holds in group $\mathbb{G}$. Thus, we conclude that for every adversary $\mathcal{A}_2$, it has a negligible probability to learn more information from the rules than the defined leakage function $\mathcal{L}_2$. The formal proof of Theorem 2 could be found in Appendix B.

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.

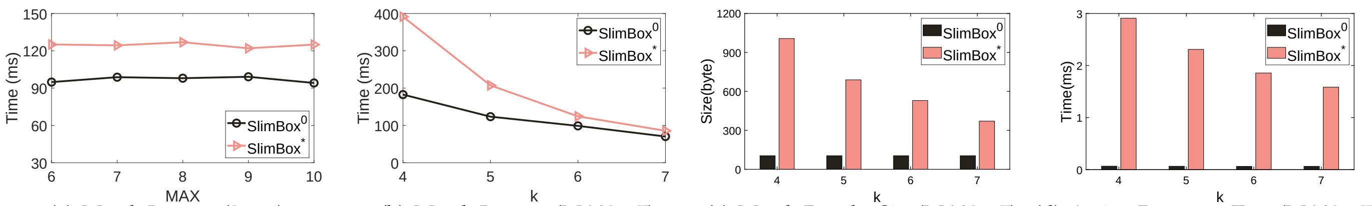
8.1 Experiment Settings and Datasets

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 λ to 256. For cryptographic algorithms, HMAC and SHA-256 are utilized to implement PRFs and collision-free



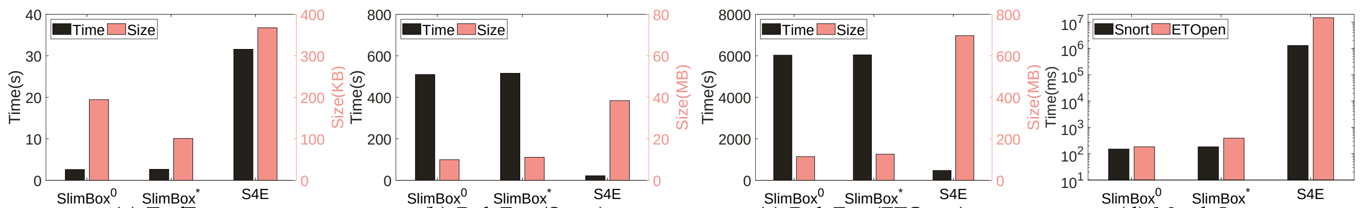
(a) Match Latency(k = 6) (b) Match Latency(MAX = 7) (c) Match Results Size(MAX = 7) (d) Action Recovery Time(MAX = 7)

Fig. 4: The results of match and action operations for SlimBox^0 and SlimBox* on Snort ruleset.



(a) Match Latency(k = 6) (b) Match Latency(MAX = 7) (c) Match Results Size(MAX = 7) (d) Action Recovery Time(MAX = 7)

Fig. 5: The results of match and action operations for SlimBox^0 and SlimBox* on ETOpen ruleset.



(a) TrafEnc (b) RuleEnc (Snort) (c) RuleEnc (ETOpen) (d) Match Latency

Fig. 6: Performance comparison with S4E on two rulesets, where $k = 4$ and $MAX = 6$ for SlimBox.

hash functions, respectively. In addition, we use AES-256 to implement SKE algorithms and use the jPBC [32] library for group operations and bilinear pairing calculations.

We choose two real open-source rulesets, Snort[3] (2502 rules, 3099 patterns) and ETOpen[4] (22516 rules, 31829 patterns) to generate the encrypted rules, and use the traffic dump iCTF08[5] to evaluate the matching performance. According to the complexity analysis, the number of offset k-grams (determined by k) and the maximum appearances k-grams MAX/MAX* (uniformly expressed by notation MAX) are two important parameters for the performance. In our experiments, we set k to $\{4, 5, 6, 7\}$, and set MAX to $\{6, 7, 8, 9, 10\}$. In addition, we choose the payloads with 1500-byte length from ICTF08 as our test traffic.

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.

3. https://www.snort.org/downloads/#rule-downloads
4. https://rules.emergingthreats.net
5. http://ictf.cs.ucsb.edu/pages/the-2008-ictf.html

• **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

TABLE 5: Performance comparison between BlindBox, S4E, and SlimBox.

Scheme	TrafEnc time (s)	Communication cost (bytes)	Inspection time
BlindBox	1.04	609630	24ms
S4E	31.94	376320	1300s
SlimBox^0	2.58	199101	158ms
SlimBox*	2.65	103293	180ms
SlimBox* (two-round)	2.65	8382	180ms

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), SlimBox* requires the additional operations (e.g., bit string decryption and hash) and thus consumes more execution time than SlimBox^0.

Comparison with prior work. We first provide the performance comparisons between S4E [19] and our SlimBox. From Fig. 6-(a), we know that SlimBox consumes less time/bandwidth in the process of traffic encryption compared with S4E. The reason is that 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 SlimBox consumes more rule encryption time than S4E. This is because SlimBox needs to encrypt each rule MAX times. However, 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 SlimBox performs much better than S4E in terms of matching latency. The huge difference between S4E and SlimBox is because S4E needs to perform expensive bilinear pairing calculations for each rule, but our SlimBox only needs to perform group operations on a fraction of rules passing inverted searching.

Then, we give comparisons between BlindBox, S4E, and our 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 Sonrt ruleset, in which the size of distinct rule patterns is equal to 91. For 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 BlindBox and S4E presented the least and the most inspection time. The main reason is that BlindBox leveraged the lightweight cryptographic primitive and thus had the optimal inspection efficiency. Our schemes are based on OXT and hence perform worse than BlindBox. As for the traffic encryption time, we can get the same comparison results as the inspection time. The BlindBox performs best due to the fast SKE, and our SlimBox also has acceptable performance. In terms of the communication cost, we can see that BlindBox and SlimBox* will incur the most and the least traffic communication cost, respectively. The reason is that BlindBox will consider all distinct rule pattern size in the traffic encryption process. While SlimBox* does not need to send cross tag set to MB compared to SlimBox hence incurs the least bandwidth. Especially, the two-round interaction SlimBox* 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 SlimBox*.

9 Conclusion

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

Acknowledgment

This work was supported in part by NSFC grants 62272150, 61872133, and 61802076; NSF grants CNS 1824440, CNS 1828363, and CNS 1757533; the Hunan Provincial Natural Science Foundation of China (Grant No. 2020JJ3015); and the Postgraduate Scientific Research Innovation Project of Hunan Province (No. QL20210095).

References

[1] Snort, "An open source intrusion prevention system," 2015. [Online]. Available: https://www.snort.org

[2] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks*, 1999.

[3] 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 SIGCOMM*, 2012.

[4] C. Wang, X. Yuan, Y. Cui, and K. Ren, "Toward secure outsourced middlebox services: practices, challenges, and beyond," *IEEE Network*, 2018.

[5] G. S. Poh, D. M. Divakaran, H. W. Lim, J. Ning, and A. Desai, "A survey of privacy-preserving techniques for encrypted traffic inspection over network middleboxes," *arXiv preprint*, 2021. [Online]. Available: https://arxiv.org/abs/2101.04338

[6] L. S. Huang, A. Rice, E. Ellingsen, and C. Jackson, "Analyzing forged SSL certificates in the wild," in *Proc. of S&P*, 2014.

[7] McAfee virtual network security platform. [Online]. Available: https://mcafee-uat.mcafee.com/enterprise/en-us/products/virtual-network-security-platform.html

[8] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Computing Surveys*, 2014.

[9] D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang, "Certificateless public key authenticated encryption with keyword search for industrial internet of things," *IEEE Transactions on Industrial Informatics*, 2018.

[10] Q. Liu, Y. Peng, S. Pei, J. Wu, T. Peng, and G. Wang, "Prime inner product encoding for effective wildcard-based multi-keyword fuzzy search," *IEEE Transactions on Services Computing*, 2022.

[11] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox: deep packet inspection over encrypted traffic," in *Proc. of SIGCOMM*, 2015.

[12] C. Lan, J. Sherry, R. A. Popa, S. Ratnasamy, and Z. Liu, "Embark: securely outsourcing middleboxes to the cloud," in *Proc. of NSDI*, 2016.

[13] S. Canard, A. Diop, N. Kheir, M. Paindavoine, and M. Sabt, "BlindIDS: market-compliant and privacy-friendly intrusion detection system over encrypted traffic," in *Proc. of ASIACCS*, 2017.

[14] X. Yuan, X. Wang, J. Lin and, C. Wang, "Privacy-preserving deep packet inspection in outsourced middleboxes," in *Proc. of INFOCOM*, 2016.

[15] J. Fan, C. Guan, K. Ren, Y. Cui, and C. Qiao, "SPABox: safeguarding privacy during deep packet inspection at a middlebox," *IEEE/ACM Transactions on Networking*, 2017.
[16] J. Ning, G. Poh, J. C. Loh, J. Chia, and E. C. Chang, "PrivDPI: privacy-preserving encrypted traffic inspection with reusable obfuscated rules," in *Proc. of CCS*, 2019.
[17] J. Ning, X. Huang, G. S. Poh, S. Xu, J. C. Loh, J. Weng, and R. H. Deng, "Pine: enabling privacy-preserving deep packet inspection on TLS with rule-hiding and fast connection establishment," in *Proc. of ESORICS*, 2020.
[18] N. Desmoulins, P. A. Fouque, C. Onete, and O. Sanders, "Pattern matching on encrypted streams," in *Proc. of ASIACRYPT*, 2018.
[19] A. Bkakria, N. Cuppens, and F. Cuppens, "Privacy-preserving pattern matching on encrypted data," in *Proc. of ASIACRYPT*, 2020.
[20] S. Lai, X. Yuan, S. Sun, J. K. Liu, S. Ron, A. Sakzad, and D. Liu, "Practical encrypted network traffic pattern matching for secure middleboxes," *IEEE Transactions on Dependable and Secure Computing*, 2021.
[21] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. "Leakage-abuse attacks against searchable encryption,". in *Proc. of CCS*, 2015.
[22] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.C. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Proc. of CRYPTO*, 2013.
[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.
[24] B. Trach, A. Krohmer, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer, "ShieldBox: secure middleboxes using shielded execution," in *Proc. of SOSR*, 2018.
[25] R. Poddar, C. Lan, R. A. Popa, and S. Ratnasamy, "Safebricks: shielding network functions in the cloud," in *Proc. of NSDI*, 2018.
[26] H. Duan, C. Wang, X. Yuan, Y. Zhou, Q. Wang, and K. Ren, "LightBox: full-stack protected stateful middlebox at lightning speed," in *Proc. of CCS*, 2019.
[27] J. Han, S. Kim, D. Cho, B. Choi, J. Ha, and D. Han, "A secure middlebox framework for enabling visibility over multiple encryption protocols," *IEEE/ACM Transactions on Networking*, 2020.
[28] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai, "SgxPectre: stealing intel secrets from sgx enclaves via speculative execution," *IEEE Security & Privacy*, 2020.
[29] J. Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: extracting the keys to the intel SGX kingdom with transient out-of-order execution," in *Proc. of USENIX Security*, 2018.
[30] X. Yuan, H. Duan, and C. Wang, "Assuring string pattern matching in outsourced middleboxes," *IEEE/ACM Transactions on Networking*, 2018.
[31] B. Choi, J. Chae, M. Jamshed, K. Park, and D. Han, "DFC: accelerating string pattern matching for network applications," in *Proc. of NSDI*, 2016.
[32] A. D. Caro, and V. Iovino, "jPBC: Java pairing based cryptography," in *Proc. of ISCC*, 2011.

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 dean 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 Networking, associate editor for the IEEE 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 CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF 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 his 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 seven consecutive years (2014-2020). 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.

Appendix A
Proof of Theorem 1

Proof: The proof proceeds through several games indistinguishable from their predecessors. The first game G_0 is structured to generate the same distribution as $\mathbf{Real}_{\mathcal{A}_1}(\lambda)$, and the last game G_6 is designed to be easily simulated by an simulator $\mathcal{S}_1$ with the predefined leakage $\mathcal{L}_1$. By relating the games, we conclude that $\mathbf{Ideal}^{\mathcal{S}_1}_{\mathcal{A}_1}(\lambda)$ is indistinguishable from $\mathbf{Real}_{\mathcal{A}_1}(\lambda)$, completing our proof.

• **Game** G_0 : This game is an implementation of the real game except that maps A, B, and R indexed by numbers $1, \ldots, n$ are used to keep the auxiliary tags and cross tags for all payload pairs. Therefore, we have:

$$\Pr[\mathbf{Real}_{\mathcal{A}_1}(\lambda) = 1] = \Pr[G_0 = 1]$$

• **Game** G_1: This game is exactly the game G_0 except that the calls to PRFs are replaced by picking random strings/numbers. G_1 maintains maps FS, FI, FZ, FV, FL, and FX for PRFs $F(K_S, \cdot)$, $F_p(K_I, \cdot)$, $F_p(K_Z, \cdot)$, $F_p(K_V, \cdot)$, $F_p(K_L, \cdot)$, and $F_p(K_X, \cdot)$, respectively. When a PRF is called, G_1 checks whether the entry in the corresponding map exists or not. If so, G_1 returns the entry; otherwise, G_1 picks a random string/number and updates the entry accordingly. By a standard hybrid argument, it is easy to show that there exists an efficient adversary $\mathcal{B}_1$ such that:

$$\Pr[G_1 = 1] - \Pr[G_0 = 1] \leq 6 \times \mathrm{Adv}^{\mathrm{prf}}_{\mathcal{B}_1}.$$

• **Game** G_2 This game replaces the action ciphertext by a random string of length λ. We claim there exists an efficient adversary $\mathcal{B}_2$ such that

$$\Pr[G_2 = 1] - \Pr[G_1 = 1] \leq m \times \mathrm{Adv}^{\mathrm{ske}}_{\mathcal{B}_2}(\lambda).$$

• **Game** G_3. This game alters the way of generating trapdoors without changing the distribution. In order to simulate trapdoors, G_3 keeps maps H_η and $\widetilde{\mathsf{H}}_\eta$ ($\eta = 1, 2$) filled as follows: For $i \in [m]$, G_3 sets $\widetilde{\mathsf{H}}_1[i] \leftarrow g^{\mathsf{FX}[\mathrm{okg}_i] \times \mathsf{FV}[\mathrm{lkg}_i]}$ and $\widetilde{\mathsf{H}}_2[i] \leftarrow g^{\mathsf{FX}[\mathrm{okg}_i] \times (\Delta_i - \mathsf{FL}[\mathrm{lkg}_i]) \times \mathsf{FZ}[\mathrm{lkg}_i]^{-1}}$. For any pair $(s, i) \in [n] \times [m]$, G_3 sets:

$$\mathsf{H}_1[s, i] \leftarrow g^{\mathsf{FI}[\mathrm{id}] \times \mathsf{FX}[\mathrm{okg}_i] \times (\mathrm{xp}_s + \mathsf{FL}[\mathrm{lkg}_i])}$$
$$\mathsf{H}_2[s, i] \leftarrow g^{\mathsf{FI}[\mathrm{id}] \times \mathsf{FX}[\mathrm{okg}_i] \times (\Delta_i - \mathsf{FL}[\mathrm{lkg}_i])}$$

For rule $R_i = (\mathrm{lkg}_i, \mathrm{okg}_i, \Delta_i)$, G_3 sets $\mathrm{ltrap}_i \leftarrow \mathsf{FS}[\mathrm{lkg}_i]$. The way to compute cross trapdoors depends on the following cases: (1) The rule does not match any payload at all. G_3 calculates $\mathrm{xtrap}_i \leftarrow \widetilde{\mathsf{H}}_1[i]$ and $\mathrm{xofs}_i \leftarrow \widetilde{\mathsf{H}}_2[i]$. (2) The rule fully/partially matches the s-th payload pair P_s, i.e., $\mathrm{lkg}_i = \mathrm{kg}_s$. G_3 computes $\mathrm{xtrap}_i \leftarrow \mathsf{H}_1[s, i]^{\mathsf{B}[s]^{-1}}$ and $\mathrm{xofs}_i \leftarrow \mathsf{H}_2[s, i]^{\mathsf{A}[s]^{-1}}$, where $\mathsf{A}[s] = \mathsf{FI}[\mathrm{id}] \times \mathsf{FZ}[\mathrm{kg}_s]$, and $\mathsf{B}[s] = \mathsf{FI}[\mathrm{id}] \times (\mathrm{xp}_s + \mathsf{FL}[\mathrm{kg}_s]) \times \mathsf{FV}[\mathrm{kg}_s]^{-1}$. For all the above cases, we have $\mathrm{xtrap}_i = g^{\mathsf{FX}[\mathrm{okg}_i] \times \mathsf{FV}[\mathrm{lkg}_i]}$ and $\mathrm{xofs}_i = g^{\mathsf{FX}[\mathrm{okg}_i] \times (\Delta_i - \mathsf{FL}[\mathrm{lkg}_i]) \times \mathsf{FZ}[\mathrm{lkg}_i]^{-1}}$, so we claim that:

$$\Pr[G_3 = 1] = \Pr[G_2 = 1]$$

• **Game** G_4. This game is exactly like G_3 except that the value of α and β are set to random values of $\mathbb{Z}_p^*$. After G_3, the value of z (resp. v) is used exactly once, and thus the value of α (resp. β) is uniform and independent of the rest of the randomness in the game. Therefore, we have:

$$\Pr[G_4 = 1] = \Pr[G_3 = 1]$$

• **Game** G_5. The main difference from G_4 is that maps R, H_η and $\widetilde{\mathsf{H}}_\eta$ for $\eta = 1, 2$ are filled with random elements from $\mathbb{G}$. The trick lies in the way to fill maps H_1 and H_2. To ensure consistency, when filling the (s, i)-th entry, G_5 first checks whether there exists a pair of non-empty entries $\mathsf{H}_1[t, j]$ and $\mathsf{H}_2[t, j]$ such that $\mathrm{okg}_j = \mathrm{okg}_i$ and $\mathrm{xp}_t + \Delta_j = \mathrm{xp}_s + \Delta_i$. If so, G_5 chooses two random elements $r_1, r_2 \in \mathbb{G}$, s.t. $r_1 \times r_2 = \mathsf{H}_1[t, j] \times \mathsf{H}_2[t, j]$, and sets $\mathsf{H}_1[s, i] \leftarrow r_1$ and $\mathsf{H}_2[s, i] \leftarrow r_2$. Otherwise, for the case that R_i fully matches P_s, G_5 locates $\mathsf{R}[t]$, s.t. $\mathrm{kg}_t = \mathrm{okg}_i$, and $\mathrm{xp}_t = \mathrm{xp}_s + \Delta_i$, chooses two random elements $r_1, r_2 \in \mathbb{G}$, s.t. $r_1 \times r_2 = \mathsf{R}[t]$, and sets $\mathsf{H}_\eta[s, i] \leftarrow r_\eta$, for $\eta = 1, 2$; For the other cases, G_5 sets $(\mathsf{H}_1[s, i], \mathsf{H}_2[s, i]) \xleftarrow{\$} \mathbb{G}$.

Let $\mathcal{B}_5$ be an adversary solving the DDH problem. Given $g^a = g^{\mathsf{FX}[\mathrm{kg}]}$, $g^b = g^{\mathsf{FI}[\mathrm{id}] \times \mathrm{xp}}$, and M, $\mathcal{B}_5$ attempts to determine $M = g^{a \times b}$ or M is selected randomly from $\mathbb{G}$. Similarly, given $g^{a'} = g^{\mathsf{FI}[\mathrm{id}]}$ and $g^{b'} = g^{\mathsf{FX}[\mathrm{kg}] \times \Delta}$, and M', $\mathcal{B}_5$ attempts to determine $M' = g^{a' \times b'}$ or M' is a random element from $\mathbb{G}$. By applying Lemma 1, we have:

$$\Pr[G_5 = 1] - \Pr[G_4 = 1] \leq \mathrm{Adv}^{\mathrm{ddh}}_{\mathcal{B}_5}(\lambda).$$

• **Game** G_6. This game makes minor changes, so that the final simulator works with the given leakage. First, G_6 omits maps H_η and $\widetilde{\mathsf{H}}_\eta$, for $\eta = 1, 2$. Instead, G_6 maintains a new map X indexed by numbers $1, \ldots, m$. Second, G_6 replaces label tags with random strings of length λ, and maintains a map T to record the label tag for each payload pair. For $i \in [m]$, if R_i does not match any payload at all, G_6 sets $\mathrm{ltrap}_i \xleftarrow{\$} \{0, 1\}^\lambda$ and $(\mathrm{xtrap}_i, \mathrm{xofs}_i) \xleftarrow{\$} \mathbb{G}$. If R_i fully/partially matches the s-th pair, G_6 calculates the trapdoors by setting $\mathrm{ltrap}_i \leftarrow \mathsf{T}[s]$, $\mathrm{xtrap}_i \leftarrow r_1^{\mathsf{B}[s]^{-1}}$, and $\mathrm{xofs}_i \leftarrow r_2^{\mathsf{A}[s]^{-1}}$, and sets $\mathsf{X}[i] \leftarrow r_1 \times r_2$. Here, $r_1, r_2 \in \mathbb{G}$ are constructed as follows: If there exists a non-empty entry $\mathsf{X}[j]$, such that $\mathrm{lkg}_j = \mathrm{kg}_t$, $\mathrm{okg}_j = \mathrm{okg}_i$, and $\mathrm{xp}_t + \Delta_j = \mathrm{xp}_s + \Delta_i$, G_6 sets $(r_1, r_2) \xleftarrow{\$} \mathbb{G}$, s.t. $r_1 + r_2 = \mathsf{X}[j]$; Otherwise, for the case that R_i fully matches P_s, G_6 sets $(r_1, r_2) \xleftarrow{\$} \mathbb{G}$, s.t. $r_1 + r_2 = \mathsf{R}[s]$; for the case that R_i partially matches P_s, G_6 sets $(r_1, r_2) \xleftarrow{\$} \mathbb{G}$. Hence, we have:

$$\Pr[G_6 = 1] = \Pr[G_5 = 1]$$

• **Simulator** $\mathcal{S}_1$. $\mathcal{S}_1$ first classifies rules into partitions according to IR, and assigns each partition with a distinct index. Let $\hat{i}$ denote the partition index of rule R_i. For two rules R_i and R_j in RS, we have $\hat{i} = \hat{j}$ iff $\mathsf{IR}[i, j] \neq 0$. With leakage $\mathcal{L}_1$, $\mathcal{S}_1$ runs Protocol 1 to generate the same distribution as G_6. The main difference from G_6 is that rule indexes are replaced by partition indexes, and thus we have:

$$\Pr[\mathbf{Ideal}^{\mathcal{S}_1}_{\mathcal{A}_1}(\lambda) = 1] = \Pr[G_6 = 1].$$

By combing all results, we can say that, for any PPT adversary $\mathcal{A}_1$, there exist adversaries $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_5$, s.t:

$$\begin{aligned} &|\Pr[\mathbf{Real}_{\mathcal{A}_1}(\lambda) = 1] - \Pr[\mathbf{Ideal}^{\mathcal{A}_1}_{\mathcal{S}_1}(\lambda) = 1]| \\ &\leq 6 \times \mathrm{Adv}^{\mathrm{prf}}_{\mathcal{B}_1}(\lambda) + m \times \mathrm{Adv}^{\mathrm{ske}}_{\mathcal{B}_2}(\lambda) + \mathrm{Adv}^{\mathrm{ddh}}_{\mathcal{B}_5}(\lambda) \end{aligned}$$

□

Protocol 1 Simulator $\mathcal{S}_1$

Simulator code to generate maps

for $s \in [n]$ **do**
    $\mathsf{T}[s] \xleftarrow{\$} \{0,1\}^\lambda$; $(\mathsf{A}[s], \mathsf{B}[s]) \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathsf{R}[s] \xleftarrow{\$} \mathbb{G}$

Simulator code to generate ET

$(\mathrm{LT}, \mathrm{XT}, \mathsf{ET}) \leftarrow \emptyset$
for $s \in [n]$ **do**
    $\mathrm{ltag} \leftarrow \mathsf{T}[s]$; $\alpha \leftarrow \mathsf{A}[s]$; $\beta \leftarrow \mathsf{B}[s]$; $\mathrm{xtag} \leftarrow \mathsf{R}[s]$
    $\mathrm{LT} \leftarrow \mathrm{LT} \cup (\mathrm{ltag}, \alpha, \beta)$; $\mathrm{XT} \leftarrow \mathrm{XT} \cup \mathrm{xtag}$
$\mathsf{ET} \leftarrow (\mathrm{LT}, \mathrm{XT})$

Simulator code to generate ERS

$(\mathsf{ERS}, \mathsf{X}) \leftarrow$ empty map
for the i-th rule pattern **do**
    if $\mathsf{MR}[i] = 0$ **then**
        $\mathrm{ltrap} \xleftarrow{\$} \{0,1\}^\lambda$; $(\mathrm{xtrap}, \mathrm{xofs}) \xleftarrow{\$} \mathbb{G}$
    if $\mathsf{MR}[i] = (1, s) \vee \mathsf{MR}[i] = (2, s)$ **then**
        $\mathrm{ltrap} \leftarrow \mathsf{T}[s]$
        if $\mathsf{X}[\hat{i}] \neq \perp$ **then**
            $r_1, r_2 \xleftarrow{\$} \mathbb{G}$, s.t. $r_1 \times r_2 = \mathsf{X}[\hat{i}]$
        else
            if $\mathsf{MR}[i] = (1, s)$ **then**
                $r_1, r_2 \xleftarrow{\$} \mathbb{G}$, s.t. $r_1 \times r_2 = \mathsf{R}[s]$
            else
                $r_1, r_2 \xleftarrow{\$} \mathbb{G}$
            $\mathsf{X}[\hat{i}] \leftarrow r_1 \times r_2$
        $\mathrm{xtrap} \leftarrow r_1^{\mathsf{B}[s]^{-1}}$; $\mathrm{xofs} \leftarrow r_2^{\mathsf{A}[s]^{-1}}$
    $\mathbf{cp} \xleftarrow{\$} \{0,1\}^\lambda$; $\mathsf{ERS}[\mathrm{ltrap}] \leftarrow (\mathrm{xtrap}, \mathrm{xofs}, \mathbf{cp})$

APPENDIX B
PROOF OF THEOREM 2

Proof: The proof is also proceeded by game hops.

• **Game** G_0 : This game is the same as the real game except that maps A and B indexed by numbers $1, \ldots, m$ are used to keep exponential values for rules. Hence, we have:

$$\Pr[\mathbf{Real}_{\mathcal{A}_2}(\lambda) = 1] = \Pr[G_0 = 1]$$

• **Game** G_1 : This game is exactly the game G_0 except that the calls to PRFs are replaced by picking random strings/numbers. G_1 maintains maps FS, FI, FZ, FV, FL, and FX for PRFs $F(K_S, \cdot)$, $F_p(K_I, \cdot)$, $F_p(K_Z, \cdot)$, $F_p(K_V, \cdot)$, $F_p(K_L, \cdot)$, and $F_p(K_X, \cdot)$, respectively. Therefore, we have:

$$\Pr[G_1 = 1] - \Pr[G_0 = 1] \leq 6 \times \mathrm{Adv}^{\mathrm{prf}}_{\mathcal{B}_1}.$$

• **Game** G_2 This game replaces the action ciphertext by a random string of length λ. We claim that:

$$\Pr[G_2 = 1] - \Pr[G_1 = 1] \leq m \times \mathrm{Adv}^{\mathrm{ske}}_{\mathcal{B}_2}(\lambda).$$

• **Game** G_3. This game alters the way of generating tags without changing the distribution. G_3 uses a map R to record the exponent of cross tag for each pair. For $s \in [n]$, G_3 sets $\mathsf{R}[s] \leftarrow \mathsf{FI}[\mathrm{id}] \times \mathsf{FX}[\mathrm{kg}_s] \times \mathrm{xp}_s$. In order to simulate tags, G_3 keeps maps H_η, and $\widetilde{\mathsf{H}}_\eta$, $(\eta = 1, 2)$ filled as follows: For $s \in [n]$, G_3 sets $\widetilde{\mathsf{H}}_1[s] \leftarrow \mathsf{FI}[\mathrm{id}] \times \mathsf{FZ}[\mathrm{kg}_s]$, and $\widetilde{\mathsf{H}}_2[s] \leftarrow \mathsf{FI}[\mathrm{id}] \times (\mathrm{xp}_s + \mathsf{FL}[\mathrm{kg}_s]) \times \mathsf{FV}[\mathrm{kg}_s]^{-1}$. For any pair $(s, i) \in [n] \times [m]$, G_3 sets:

$$\mathsf{H}_1[s,i] \leftarrow \mathsf{FI}[\mathrm{id}] \times \mathsf{FX}[\mathrm{okg}_i] \times (\Delta_i - \mathsf{FL}[\mathrm{lkg}_i])$$
$$\mathsf{H}_2[s,i] \leftarrow \mathsf{FI}[\mathrm{id}] \times \mathsf{FX}[\mathrm{okg}_i] \times (\mathrm{xp}_s + \mathsf{FL}[\mathrm{lkg}_i])$$

For the s-th payload $P_s = (\mathrm{kg}_s, \mathrm{xp}_s)$, G_3 calculates $\mathrm{ltag}_s \leftarrow \mathsf{FS}[\mathrm{kg}_s]$ and $\mathrm{xtag}_s \leftarrow g^{\mathsf{R}[s]}$. The way to compute auxiliary tags depends on the following cases: (1) The payload does not match any rule at all. G_3 calculates $\alpha \leftarrow \widetilde{\mathsf{H}}_1[s]$ and $\beta \leftarrow \widetilde{\mathsf{H}}_2[s]$. (2) The payload fully/partially matches rule R_i, i.e., $\mathrm{lkg}_i = \mathrm{kg}_s$. G_3 calculates $\alpha \leftarrow \mathsf{H}_1[s,i] \times \mathsf{B}[i]^{-1}$, and $\beta \leftarrow \mathsf{H}_2[s,i] \times \mathsf{A}[i]^{-1}$, where $\mathsf{A}[i] \leftarrow \mathsf{FX}[\mathrm{okg}_i] \times \mathsf{FV}[\mathrm{lkg}_i]$ and $\mathsf{B}[i] \leftarrow \mathsf{FX}[\mathrm{okg}_i] \times (\Delta_i - \mathsf{FL}[\mathrm{lkg}_i]) \times \mathsf{FZ}[\mathrm{lkg}_i]^{-1}$. For all the above cases, the tags are generated in the same form. Therefore, we have:

$$\Pr[G_3 = 1] = \Pr[G_2 = 1]$$

• **Game** G_4. This game is exactly like G_3 except that the value of A and B are set to random values of $\mathbb{Z}_p^*$. After G_3, the value of z (resp. v) is used exactly once, and thus the value of $\mathsf{A}[i]$ (resp. $\mathsf{B}[i]$) is uniform and independent of the rest of the randomness in the game. Therefore, we have:

$$\Pr[G_4 = 1] = \Pr[G_3 = 1]$$

• **Game** G_5. The main difference from G_4 is that maps R, H_η, and $\widetilde{\mathsf{H}}_\eta$ for $\eta = 1, 2$ are filled with random elements from $\mathbb{Z}_p^*$. The trick lies in the way to fill maps R and H_η. To ensure consistency, when filling the (s, i)-the entry, G_5 first checks whether there exist a pair of non-empty entries $\mathsf{H}_1[t, j]$ and $\mathsf{H}_2[t, j]$ such that $\mathrm{okg}_j = \mathrm{okg}_i$ and $\mathrm{xp}_t + \Delta_j = \mathrm{xp}_s + \Delta_i$. If so, G_5 chooses two random elements $r_1, r_2 \in \mathbb{Z}_p^*$, s.t. $r_1 + r_2 = \mathsf{H}_1[t, j] + \mathsf{H}_2[t, j]$ and sets $\mathsf{H}_\eta[s, i] \leftarrow r_\eta$ for $\eta = 1, 2$. For $i \in [s]$, if P_s fully matches R_i, G_5 locates R_j s.t. $\mathrm{kg}_j = \mathrm{okg}_i$ and $\mathrm{xp}_j = \mathrm{xp}_s + \Delta_i$, and sets $\mathsf{R}[j] \leftarrow \mathsf{H}_1[s, i] + \mathsf{H}_2[s, i]$. For all the remaining entries, the values are set to random elements from $\mathbb{Z}_p^*$. Let $\mathcal{B}_5$ be an adversary solving the DDH problem. By applying Lemma 1, we have:

$$\Pr[G_5 = 1] - \Pr[G_4 = 1] \leq \mathrm{Adv}^{\mathrm{ddh}}_{\mathcal{B}_5}(\lambda).$$

• **Game** G_6. To let the final simulator work with the given leakage, this game makes following minor changes: First, G_6 omits maps R, H_η, and $\widetilde{\mathsf{H}}_\eta$, for $\eta = 1, 2$. Instead, G_6 maintains a map X to record the exponent of each calculated cross tag. For $i \in [m]$, if rule R_i fully/partially matches payload P_s, G_6 checks whether there exists a rule R_j that fully/partially matches payload P_j, s.t. $\mathrm{okg}_j = \mathrm{okg}_i$ and $\mathrm{xp}_t + \Delta_j = \mathrm{xp}_s + \Delta_i$. If so, G_6 makes sure that $\mathsf{X}[i]$ and $\mathsf{X}[j]$ are filled with the same value from $\mathbb{Z}_p^*$. Otherwise, G_6 fills $\mathsf{X}[i]$ with a random value from $\mathbb{Z}_p^*$. Second, G_6 maintains a map F indexed by numbers $1, \ldots, m$, and initializes each entry with 0. Third, G_6 replaces label trapdoors with random strings of length λ, and maintains a map T to record the label trapdoor for each rule. For $s \in [n]$, if payload P_s does not match any rule at all, G_6 sets $\mathrm{ltag}_s \xleftarrow{\$} \{0,1\}^\lambda$,

Protocol 2 Simulator $\mathcal{S}_2$

Simulator code to generate maps

for $j \in \mathbf{p}$ **do**
  $\mathsf{X}[j] \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathsf{F}[j] \leftarrow 0$
for $i \in [m]$ **do**
  $\mathsf{T}[i] \xleftarrow{\$} \{0,1\}^\lambda$; $(\mathsf{A}[i], \mathsf{B}[i]) \xleftarrow{\$} \mathbb{Z}_p^*$

Simulator code to generate ERS

$\mathsf{ERS} \leftarrow$ empty map
for $i \in [m]$ **do**
  $\mathrm{ltrap} \leftarrow \mathsf{T}[i]$; $\mathrm{xtrap} \leftarrow g^{\mathsf{A}[i]}$; $\mathrm{xofs} \leftarrow g^{\mathsf{B}[i]}$
  $\mathbf{cp} \xleftarrow{\$} \{0,1\}^\lambda$; $\mathsf{ERS}[\mathrm{ltrap}] \leftarrow (\mathrm{xtrap}, \mathrm{xofs}, \mathbf{cp})$

Simulator code to generate ET

$(\mathrm{LT}, \mathrm{XT}) \leftarrow \emptyset$; $\mathsf{ET} \leftarrow$ empty map
for $s \in [n]$ **do**
  if $\mathsf{MR}[s] = (1, i) \vee \mathsf{MR}[s] = (2, i)$ **then**
    $\mathrm{ltag} \leftarrow \mathsf{T}[i]$
    $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$ s.t. $r_1 + r_2 = \mathsf{X}[\hat{i}]$
    $\alpha \leftarrow r_1 \times \mathsf{B}[i]^{-1}$; $\beta \leftarrow r_2 \times \mathsf{A}[i]^{-1}$
    if $\mathsf{MR}[s] = (1, i) \wedge \mathsf{F}[\hat{i}] = 0$ **then**
      $\mathrm{xtag} \leftarrow g^{\mathsf{X}[\hat{i}]}$; $\mathsf{F}[\hat{i}] \leftarrow 1$
    else
      $\mathrm{xtag} \xleftarrow{\$} \mathbb{G}$
  if $\mathsf{MR}[s] = 0$ **then**
    $\mathrm{ltag} \xleftarrow{\$} \{0,1\}^\lambda$; $(\alpha, \beta) \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathrm{xtag} \xleftarrow{\$} \mathbb{G}$
  $\mathrm{LT} \leftarrow \mathrm{LT} \cup (\mathrm{ltag}, \alpha, \beta)$; $\mathrm{XT} \leftarrow \mathrm{XT} \cup \mathrm{xtag}$
$\mathsf{ET} \leftarrow (\mathrm{LT}, \mathrm{XT})$

$(\alpha_s, \beta_s) \xleftarrow{\$} \mathbb{Z}_p^*$, and $\mathrm{xtag}_s \xleftarrow{\$} \mathbb{G}$. If P_s fully/partiall matches R_i, G_5 sets the label tag as $\mathrm{ltag}_s \leftarrow \mathsf{T}[i]$, and the auxiliary tags as $\alpha_s \leftarrow r_1 \times \mathsf{B}^{-1}[i]$, $\beta_s \leftarrow r_2 \times \mathsf{A}^{-1}[i]$, where r_1, r_2 are two random elements from $\mathbb{Z}_p^*$ s.t. $r_1 + r_2 = \mathsf{X}[i]$. For the case that P_s fully matches R_i and $\mathsf{F}[i] = 0$, G_6 calculates the cross tag by setting $\mathrm{xtag}_s \leftarrow \mathsf{X}[i]$ and updates the map by $\mathsf{F}[i] \leftarrow 1$. Furthermore, for $j \in [m]$, if $\mathsf{X}[j] = \mathsf{X}[i]$, G_6 sets $\mathsf{F}[j] \leftarrow 1$. Otherwise, G_6 sets $\mathrm{xtag}_s \xleftarrow{\$} \mathbb{G}$. Hence, we have:

$$\Pr[G_6 = 1] = \Pr[G_5 = 1]$$

• **Simulator** $\mathcal{S}_2$ $\mathcal{S}_2$ first classifies rules into partitions according to IR, and assigns each partition a distinct index. Let $\hat{i}$ denote the partition index of rule R_i, and let $\mathbf{p}$ denote the set of partition indexes. With leakage $\mathcal{L}_2$, $\mathcal{S}_2$ runs Protocol 2 to generate the same distribution as G_6. The main difference from G_6 is that rule indexes are replaced by partition indexes, and thus we have:

$$\Pr[\mathbf{Ideal}_{\mathcal{A}_2}^{\mathcal{S}_2}(\lambda) = 1] = \Pr[G_6 = 1].$$

By combing all results, we can say that, for any PPT adversary $\mathcal{A}_2$, there exist adversaries $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_5$ s.t.:

$$\begin{aligned} &|\Pr[\mathbf{Real}_{\mathcal{A}_2}(\lambda) = 1] - \Pr[\mathbf{Ideal}_{\mathcal{S}_2}^{\mathcal{A}_2}(\lambda) = 1]| \\ &\leq 6 \times \mathrm{Adv}_{\mathcal{B}_1}^{\mathrm{prf}}(\lambda) + m \times \mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{ske}}(\lambda) + \mathrm{Adv}_{\mathcal{B}_5}^{\mathrm{ddh}}(\lambda) \end{aligned}$$

$\square$