Hindawi
Security and Communication Networks
Volume 2022, Article ID 4743078, 13 pages
https://doi.org/10.1155/2022/4743078

Research Article

Encrypted Packet Inspection Based on Oblivious Transfer

Xi Jia[1] and Meng Zhang[1,2]

[1]*College of Computer Science and Technology, Jilin University, Changchun, China*
[2]*Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, China*

Correspondence should be addressed to Meng Zhang; zhangmeng@jlu.edu.cn

Received 14 December 2021; Revised 19 April 2022; Accepted 27 May 2022; Published 24 August 2022

Academic Editor: Youwen Zhu

Deep packet inspection (DPI) is widely used in detecting abnormal traffic and suspicious activities in networks. With the growing popularity of secure hypertext transfer protocol (HyperText Transfer Protocol over Secure Socket Layer, HTTPS), inspecting the encrypted traffic is necessary. The traditional decryption-and-then-encryption method has the drawback of privacy leaking. Decrypting encrypted packets for inspection violates the confidentiality goal of HTTPS. Now, people are faced with a dilemma: choosing between the middlebox's ability to perform detection functions and protecting the privacy of their communications. We propose OTEPI, a system that simultaneously provides both of those properties. The approach of OTEPI is to perform the deep packet inspection directly on the encrypted traffic. Unlike machine and deep learning methods that can only classify traffic, OTEPI is able to accurately identify which detection rule was matched by the encrypted packet. It can facilitate network managers to manage their networks at a finer granularity. OTEPI achieves the function through a new protocol and new encryption schemes. Compared with previous works, our approach achieves rule encryption with oblivious transfer (OT), which allows our work to achieve a better balance between communication traffic consumption and computational resource consumption. And our design of Oblivious Transfer and the use of Natural Language Processing tools make OTEPI outstanding in terms of computational consumption.

1. Introduction

Packet inspection and analysis are widely used to detect, mitigate, and prevent suspicious network activities. Real-time inspection of packet payloads and headers is essential to achieve these goals. The equipment deployed for these purposes is middlebox, an intermediate device providing various services. Middlebox is essential in today's network infrastructure. The primary services provided by middlebox include firewalls, intrusion detection, parental filtering, data leakage detection, forensic analysis, malware analysis, and others.

With the popularity of HTTPS, 87%–90% of the current network traffic is encrypted by protocols such as SSL (Secure Sockets Layer) and TLS (Transport Layer Security)[1]. According to Google's statistics, in November 2020, 81% to 98% of the traffic from the Chrome browser on different platforms used HTTPS [2]. The Man-in-the-Middle (MitM) technology is one of the commonly used approaches to inspect encrypted traffic. An MB establishes encrypted connections between both the client and the server. The middlebox decrypts the traffic in the connections, inspects the payloads according to the intrusion detection rules, and then re-encrypts the payloads. When rules are matched, alerts will be sent to the administrator to take actions such as disconnection and alert.

This decrypt-and-detect approach violates the security goal of HTTPS. A survey [3] from USENIX Association shows that 75.8% of users have privacy concerns about the MitM system that decrypts encrypted traffic, and 83.2% of the surveyors believe that the third-party inspection should be notified in advance. The MitM technology can achieve either the function of traffic inspection or privacy in communication due to the intrinsic conflict between the two.

With the popularity of HTTPS, TLS/SSL encrypted traffic in the network has gradually become the majority. Performing the traffic inspection while protecting the privacy of both parties has become a problem that attracted much research. We hope to implement a traffic inspection that provides privacy protection, propose a new method to optimize the bandwidth, and overhead compared with the previous methods.

1.1. Related Works. Encrypted traffic detection technologies are classified into three categories: Searchable Encryption, Machine and Deep Learning, and Trusted Hardware. We give a brief survey as follows.

1.1.1. Searchable Encryption. Sherry et al. proposed the BlindBox [4] in 2015, which is the first privacy-preserving deep packet inspection scheme using searchable encryption techniques. They adopted oblivious transfer (OT) [5–7] along with garbled circuits (GC) [8, 9] to perform the inspection of the encrypted traffic without decrypting the payloads. The middlebox cannot access the plaintext in the traffic, and the client and the server cannot learn the content of the rules. While this method achieved the desired privacy protection, it requires a significant amount of calculation and communication due to the use of garbled circuits [10]. In addition to the considerable overhead of the garbled circuit itself, every new rule in each new session has to generate a new garbled circuit.

To address the performance bottleneck of the BlindBox, Canard et al. proposed BlindIDS [11], which is a token-matching protocol that uses a Decryptable Searchable Encryption (DSE) tool based on the pairing-based public key algorithm. Compared with BlindBox, BlindIDS drastically reduce the overhead of the rules setup, moving some overhead to the middlebox detection phase.

Fan et al. introduced the SPABox [12], which uses oblivious pseudorandom functions in the rule encryption. The middlebox portion performs two matching operations, namely, keyword matching and machine learning model matching. Like all searchable encryption methods, SPABox also adopts tokenization. The only difference between the SPABox and other methods is the adoption of machine learning which uses a different approach in token matching.

Ning et al. proposed the PrivDPI [13]. This method improves the BlindBox that enhances the setup phase to reduce bandwidth consumption. They introduced the idea of reusable encryption rules, which significantly reduced the bandwidth overhead in the case of multiple continuous sessions. However, modular operations in token encryption increase the computational overhead compared with BlindBox.

1.1.2. Machine and Deep Learning. Machine learning technology is also widely used in encrypted traffic inspection. Yamada et al. [14] proposed a new anomaly detection technology. This method performs anomaly detection by analyzing the data packet's size and the flow's temporal characteristics. The scheme proposed by Anderson et al. [15, 16] detects malicious programs mainly by TLS header information and DNS data. The article finds that encrypted malware traffic has different characteristics from regular traffic.

Deep learning methods are also widely used in intrusion detection for network security. Ferrag et al. [17] analyzed the performance of seven deep learning models is analyzed for three metrics: accuracy, false alarm rate, and detection rate under different data sets. Montieri et al. [18] proposed a scheme that allows traffic classification in anonymous browsers (e.g., tor) employing a hierarchical approach. In detail, the proposed framework was designed with varying constraints, resulting in implementations with different degrees of complexity (in terms of classifiers, features, and reject options). Adept [19] is an attack detection and identification framework for identifying multi-stage distributed attacks on the Internet-of-Things (IoT). It is based on a hierarchical distributed framework, where local gateways monitor network traffic and generate alerts for any anomalous activity. The central security manager detects attacks by mining the aggregated alerts and identifies corresponding attack stages using a comprehensive set of features via machine learning. Liu et al. [20] proposed a Flow Sequence Network (FS-Net) scheme that uses recurrent neural networks for encrypted traffic classification. The FS-Net takes a multi-layer bi-GRU [21] encoder to learn the representation of the flow sequence and reconstructs the original sequence with a multi-layer bi-GRU decoder. The features learned from the encoder and decoder are combined for classification. Aceto et al. [22] proposed a traffic classifier called DISTILLER, a multi-modal multi-task deep learning method. DISTILLER addresses the performance limitations of existing traffic classifiers based on single-mode deep learning and provides an adequate design basis for sophisticated network management requiring the solution of different network visibility tasks. A new method for classifying end-to-end encrypted traffic using one-dimensional convolution neural networks is proposed by Wang et al. [23] based on the analysis of traditional methods for classifying encrypted traffic in machine learning. This strategy differs from the traditional divide-and-conquer strategy. The 1D-CNN classification strategy integrates feature design, feature extraction, and feature selection in a single framework, making it more likely to yield a globally optimal solution than a divide-and-conquer strategy. A Tree-Shaped Deep Neural Network (TSDNN) and Quantity Dependent Backpropagation (QDBP) algorithm was proposed by Chen et al. [24]. This model can memorize and learn from the minority class, to perform malicious flow detection.

1.1.3. Trusted Hardware. Trusted hardware is a new technology that has been deployed for privacy-preserving deep packet inspection. David Goltzsche et al. presented ENDBox [25] that uses the Intel SGX tool provided by Intel, which supports both local and remote verification. ENDBox and SGX are connected through a virtual network, and the traffic is decrypted and inspected inside the SGX. The SGX between

the two terminals is directly connected to enable the direct transmission of the traffic, without which the SGX traffic will be garbled. The McTLS proposed by David Naylor et al. [10] modifies the existing TLS protocol and use the SGX to allow the client, middlebox, and the server to establish an authenticated and secure channel to exchange read and write secret keys in addition to the session key.

1.1.4. Summary. The searchable encryption-based scheme can enhance privacy protection, but it will incur a huge overhead. All searchable encryptions require two data flows: the TLS session and the tokenized data. Although most of the works performing traffic and malware classification leverage Machine and Deep Learning Approaches. However, machine and deep learning methods extensively depend on reliable training sets. And this method can only classify the traffic and cannot accurately identify the exact rule matched. Furthermore, in Trusted Hardware technology, the security of the Trusted Hardware (i.e., Intel SGX) is still actively being studied. Moreover, it is less efficient for inspection, at least when compared to the searchable encryption schemes discussed. Therefore, improving the efficiency of the searchable encryption method is still a meaningful research direction.

1.2. Our Contributions. This paper proposes an encrypted packet inspection scheme based on oblivious transfer, namely, OTEPI (Encrypted Packet Inspection Based on Oblivious Transfer). OTEPI is in the category of searchable encryption-based scheme. OTEPI reduces the bandwidth consumption required to rules encryption without increasing the cost at the packet sender compared to BlindBox. We also adopt the idea of reusable encryption rules in PrivDPI. Though the bandwidth consumption is higher than that of PrivDPI, the computational costs of the packet sender is less than that of the PrivDPI. In general, for arbitrary types of data, the proposed scheme is able to strike a balance between the low computational resource consumption and high bandwidth consumption of BlindBox and the low bandwidth consumption and high computational resource consumption of PrivDPI. In particular, for plaintext data such as HTML web pages, our scheme optimizes the tokenization method, which is smaller than either PrivDPI or BlindBox in terms of computational resource consumption.

Table 1 shows the specific characteristics of our proposed OTEPI method. OTEPI uses oblivious transfer to reduce bandwidth overhead and does not use the exponential operation to reduce computing overhead. We also use NLP tools to segment tokens to reduce the number of tokens.

Our contributions are as follows:

(1) We designed a new rule encryption method based on oblivious transfer that can protect the privacy of both the traffic and rules and realizes the reuse of encryption rules. Compared to BlindBox, the rule encryption consumes much less bandwidth; the bandwidth required for 3000 rules encryption is reduced from 50 GB in BlindBox to 82 MB.

(2) We designed a new rule encryption method based on oblivious transfer that can protect the privacy of traffic and rules and realize the reuse of encryption rules. The tokenization reduces the number of tokens compared to the sliding window method. We only generate 10% to 20% of the tokens generated by BlindBox. Our encryption performance with NLP is 1.7 times faster than BlindBox and 7.6 times faster than PrivDPI. For recurring packets, our token encryption is 3.5 times more efficient than BlindBox and 3.8 times more efficient than PrivDPI.

(3) We use the sliding window tokenization for payloads unsuitable for NLP, such as images and audio. In this case, token encryption of OTEPI is 2.6 times slower than that of PrivDPI. For recurring packets, OTEPI is more efficient than PrivDPI. Although it is not as efficient as BlindBox in encryption, OTEPI consumes less bandwidth than BlindBox.

1.3. Article Structure. We organize the paper as follows. Section 1 reviews the related work and presents the contributions of this paper. Section 2 describes the system architecture, threat model, and preliminaries. Section 3 details our scheme. Section 4 provides correctness and security analysis. Section 5 gives the performance evaluations. We conclude in Section 6.

2. Overview

We provide notations, the system architecture, and the threat model used in the paper.

2.1. Preliminaries. For a vector or 1-D array P, we use P_i to denote the i-th element of P. For a matrix or 2-D array Q, the entry in the i-th row and j-th column is denoted by $Q_{i,j}$, the i-th row vector of Q is denoted by Q_i. For a bit string $s = s_1 s_2 \cdots s_m$, s_j denotes the j-th bit of s. As in BlindBox and PrivDPI, we tokenize the network traffic to a series of tokens, and the lengths of rules and tokens are fixed to m bits.

2.1.1. Oblivious Transfer. We define the 1-out-of-2 oblivious transfer protocol between two parties, **A** and **B**. **B** has twobit-strings $\mathbf{D}_0$ and $\mathbf{D}_1$. **A** has a bit b. When the protocol completes, **A** gets $\mathbf{D}_b$ without knowing $\mathbf{D}_{1-b}$, and **B** has no information on b. The process is denoted as $OT(\{\mathbf{A}, b\}, \{\mathbf{B}, \mathbf{D}_0, \mathbf{D}_1\})$. We build the oblivious transfer based on the Even-Goldreich-Lempel OT protocol proposed by Even et al. [5].

In this paper,$\mathrm{Enc}_{pk}(\cdot)$ represents the public key encryption and $\mathrm{Dec}_{sk}(\cdot)$ represents the private key decryption.

(1) For each OT, parties **A** and **B** first share two m-bit string y_0 and y_1.

(2) A selects an m-bit string x and the input bit b, computes the ζ to send to **B**.

$$\zeta = \mathrm{Enc}_{pk_B}(x) - y_b. \tag{1}$$

Table 1: Characteristics of existing schemes.

Characteristic	BlindBox	PrivDPI	OTEPI
Bandwidth consumption	High	Low	Mid
Encryption time consumption	Mid	High	Low
Encryption rule calculation method	GC and OT	ECC	OT
Segmentation method of tokens	Sliding window	Sliding window	NLP

(3) **B** calculates as follows and sends to **A**

$$z_0 = \mathrm{Dec}_{sk_B}(\zeta + y_0) + \mathbf{D}_0, \tag{2}$$

$$z_1 = \mathrm{Dec}_{sk_B}(\zeta + y_1) + \mathbf{D}_1. \tag{3}$$

(4) **A** as the receiver of oblivious transfer, the formula for calculating $\mathbf{D}_b$ is as follows:

$$\mathbf{D}_b = z_b - x. \tag{4}$$

2.2. The Architecture. Our solution has a similar architecture to BlindBox [4]. Its architecture is shown in Figure 1.

The system consists of four entities: Rule Generator (RG), middlebox (MB), the client (C), and server (S). RG is a third-party agency generating rules. MB monitors the traffic using the rules provided by RG. C is the party sending network traffic. S is the party that receives the traffic sent by C.

The client encrypts tokens with the secret key shared by client and server in the setup phase. Moreover, MB encrypts the rules with this secret key. MB only needs to check whether the encryption rules and the encrypted token are the same, and there is no need to decrypt the payloads or tokens.

The goal of the system is that the MB can detect the matching of rules in traffic and cannot access the plaintext of encrypted traffic and the rules from RG; the client and server cannot access the rules.

2.3. Threat Model. We assume that at least one of the client and server in a session is honest, and MB is semi-honest (honest but curious). This assumption is the same as these in BlindBox [4] and PrivDPI [13]. Under this security assumption, there are two threats. The first comes from either S or C. One of S and C can be malicious, but two entities will not be malicious simultaneously. The cases when both C and S are malicious are beyond the scope of our assumptions because they can deceive the MB by collusion.

The second threat comes from the MB. MB will not actively attack encrypted payloads but will monitor and analyze encrypted tokens to learn the content of the encrypted traffic.

3. Encrypted Traffic Inspection by Oblivious Transfer

This section introduces our oblivious transfer-based encrypted traffic inspection approach. The approach uses the same system architecture and threat model as BlindBox and PrivDPI. Unlike BlindBox and PrivDPI, we use OT solely to achieve secure multi-party computation [26] to encrypt rules. Meanwhile, to address the problem of excessively useless tokens generated by current tokenize methods, we introduce an NLP-based tokenizing method, which significantly reduces the number of tokens.

Table 2 describes the variables used in our approach.

3.1. System Flow. Our solution includes the following phases:

(1) Setup: MB receives rules and rule validation from RG.

(2) Rule preprocessing: MB interacts with C and S to establish a set of reusable obfuscated rules using oblivious transfer. It ensures that C and S will not learn the rules, and MB cannot learn the key used by C and S.

(3) Packet tokenization and token encryption: C tokenizes the payloads, encrypts the tokens, and loads them into traffic.

(4) Token inspection: MB inspects the tokens sent by C to search for the matching of the rules obtained in equation (2).

(5) Token validation: S checks the whether the tokens sent by C accord with the payloads of the TLS/SSL session.

3.2. Setup. The ruleset from RG is denoted by$R = \{R_1), \ldots, (R_n\}$. In this phase, C, S, and MB set up the parameters used in the procedure. We assume that MB has the public key of RG, and C and S have the public key of MB. For each R_i, RG generates a rule verification pair: $(X_i, \mathrm{sign}(X_i))$, where $X_i$is the ciphertext of R_i encrypted by the public key of MB, $\mathrm{sign}(X_i)$ is the signature of X_i signed by RG. RG sends all rule verification pairs $(X_i, \mathrm{sign}(X_i))$ to MB. MB gets the public key of RG and decrypts each X_i to have the rule set.

Next, C and S establish a session with a session key *sk*. C and S generate K, e, and w using the same method with *sk* as a random seed. Array K has m entries where each entry is a pair of m-bit strings. $K_j, 1 \le j \le m$ denotes the j-th pair of K, and $K_j^b, b \in \{0, 1\}$ is the b-th bit-string of the pair K_j. K is used in token and rule encryption by OT. In OT, K_j^b serves as the stand-in for the b bit for the bit on position j of the token. K has a total of $2\,\mathrm{m}^2$ bits.

The random number w is shared by C, S, and MB, which is used as the seed to generate a random array W. This array will be employed to confuse duplicate tokens against frequency-based attacks by MB. We give the whole process in Algorithm 1.

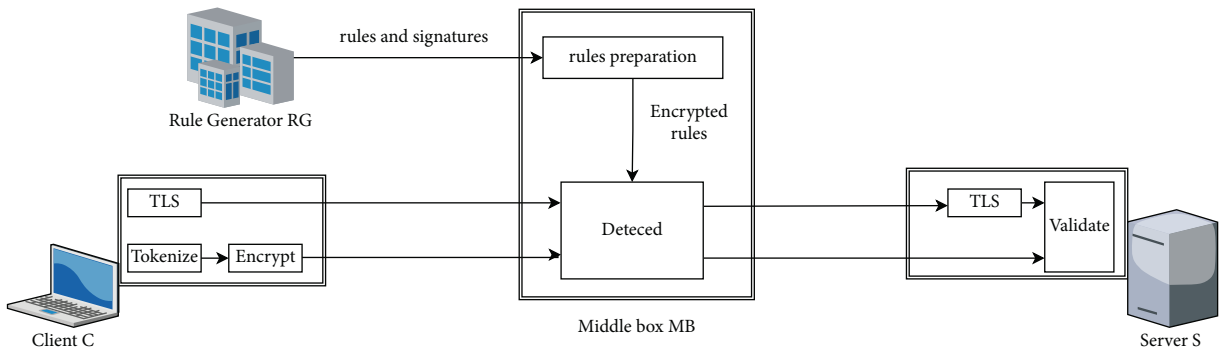

Figure 1: System architecture.

Table 2: Variables.

Name	Description	Type	Known by
n	The number of rules	int	MB, C, S
m	The number of bits in a token/rule	int	MB, C, S
e	Confusion parameter on the C/S side	m-bit string	C, S
R	The rule set, where R_i is the i-th rule	Each R_i is an m-bit string	MB
K	Array of keys of C/S to encrypt tokens and rules	Array of m pairs, each pair has two m bit strings	C, S
KM	Mask vector for obfuscating K used in OT	$n \times m$ array of m-bit strings	C, S
KE	Encrypted K by KM	$n \times m$ array of m-bit strings	C, S
RK	The keys to encrypt rules worked out by MB via OT with C, S	Array of m-bit strings of n elements	MB
RE	Encrypted rules	Array of m-bit strings of n elements	MB
W_t	Masks to obfuscate the t-th occurrence of a token	m-bit string	C, S, MB

RG:
for $i \leftarrow 1$ to n **do**
  $X_i \leftarrow$ encrypts R_i with MB's public key.
  $\text{sign}(X_i) \leftarrow$ signs X_i with RG's private key.
  sends $(X_i, \text{sign}(X_i))$ to MB.
end for
C/S:
uses sk as a random seed to generate K, e, and w.
generate two random big prime numbers ρ, σ.
$\eta \leftarrow \rho \cdot \sigma$.
send w and η to MB on a secure channel.
MB:
decrypts and verifies each R_i from $(X_i, \text{sign}(X_i))$.

Algorithm 1: Setup step.

$\text{Enc}(p, t)$ represents the encryption result of the t-th occurrence of the string p. $F(\cdot)$ denotes a one-way function. In this paper, we use the Rabin one-way function. C is responsible for generating the parameters of the Rabin function, including two big prime numbers ρ, σ, and $\eta = \rho \cdot \sigma$. C sends η to MB and S and keeps ρ, σ secret.

3.3. Rule Preprocessing. In this phase, MB will obtain the rule set encrypted with K by oblivious transfers with C and S. The procedure of the rule preprocessing is shown in Figure 2.

The security requirement in the rule encryption procedure is that MB should not obtain the key array K, and C/S should not obtain any rule. The rule encryption process has the following steps (a)–(e).

MB processes the rule set as follows:

(a) Standardization of rule length: MB pads the each R_i with 0's or computes the hash value such that the resulting length $8\lceil |R_i|/8 \rceil$. Each R_i is a string of m-bit length.

C/S performs the following operations:

(b) Verification: C/S uses the public key of RG to check whether $\text{sign}(X_i)$ and X_i are matched.

(c) Generate the confusion vector of the key: for each R_i, C/S generates a mask array KM_i that is an array of m entries and $KM_{i,j}$, $j \in [1, m]$ is an m-bit string. For any KM_i, the following relation holds:

$$e = \oplus_{j=1}^{m} KM_{i,j}. \tag{5}$$

Then, C/S encrypts K using KM. The result is an array, namely, KE, where

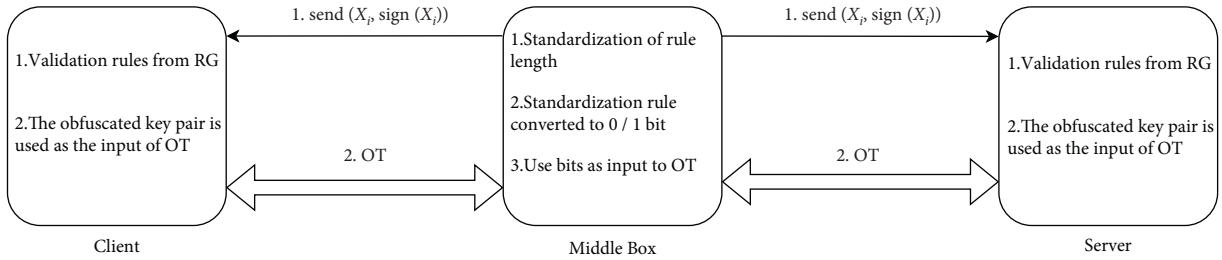

Figure 2: Rule preprocessing outline.

$$KE_{i,j}^{b} = K_j^b \oplus KM_{i,j}, \text{for } i, j \in [1, m], b \in \{0, 1\}. \quad (6)$$

Next, MB and C/S run the OT as follows.

(e) Rule encryption: for each bit $(R_i)_j$, MB and C run the OT protocol as follows, OT({MB, $(R_i)_j$}, $\{C, KE_{i,j}^0, KE_{i,j}^1\}$).

By the OT, MB gets all $KE_{i,j}^{(R_i)_j}$ from C/S. MB then computes the keys used to encrypt the rule set, namely, RK. RK is an array of m entries, where

$$RK_i = \oplus_{j=1}^{m} KE_{i,j}^{(R_i)_j}. \quad (7)$$

Bit-string RK_i is the key used in encrypting R_i.

MB and S run the same procedure, and MB computes an alternative RK. MB checks whether the two RK's are the same. If the results are different, the procedure stops. Finally, MB encrypts each R_i as follows:

$$RE_i = \overleftarrow{F}(RK_i \oplus R_i). \quad (8)$$

The whole process is shown in Figure 3. The mask KM prevents MB from revealing the content of K, but also allows MB to encrypt the rule set with K. We will prove the correctness in Section 4.2.

3.3.1. Obfuscating Repeated Tokens. We use an array W to hide the repeated tokens, an array of m-bit strings. MB and C/S use the same method to generate the same W. A token with t copies in previous tokens will be masked by W_t; thus, all encrypted tokens of the same token will be different. We set W_0 to m 0 bits.

3.4. Packet Tokenization. We introduce natural language processing (NLP) to traffic tokenization. Many packets have a text payload of natural languages and program codes. These texts are composed of words (keywords) and delimiters used to represent the grammatical structure of the text. The inspection rules for these texts also have the same proprieties, such as the parent control system and keyword censoring. The NLP-based tokenization segments the token without generating tokens that violate the structure properties. The NLP-based tokenization also supports languages with longer encoding, such as Chinese, Japanese, and Korean. C pads each token with 0's or computes the hash value as MB does with rules, which ensures that both the tokens and the rules are m-bit strings.

Protocol 1 Rule preprocessing protocol

Input: *MB* inputs (R_i), C/S inputs (K, e)

Protocol:

1. *MB*: standardizes the R_i length.
2. C/S: for each R_i generate a random mask vector array KM_i.
3. for j = 1 to m do
   (a) *C/S*: $KE_{i,j}^0 \leftarrow K_j^0 \oplus KM_{i,j}$, $KE_{i,j}^1 \leftarrow K_j^1 \oplus KM_{i,j}$.
   (b) *OT* ({C/S, $(R_i)_j$}, {MB, $KE_{i,j}^0$, $KE_{i,j}^1$}).
   end for
4. *MB*: computes $RK_i = KE_{i,j}^{(R_i)_1} \oplus \ldots \oplus KE_{i,m}^{(R_i)_m}$.
5. *MB*: Determine whether the computed encryption rules are correct:

Case 1: If the RK_i calculation results obtained from the C/S side are the same, *MB* calculates the encryption rules $RE_i \leftarrow \overleftarrow{F}(RK_i \oplus R_i)$ used for matching tokens.

Case 2: If the RK_i calculation results obtained from the C/S side are different, *MB* knows that at least one of two parties is malicious.

Figure 3: Rule preprocessing protocol.

3.5. Token Encryption. We take a token as an m-bit bit-string. Given a token of content p such that there are t tokens with content p in previous tokens, the C encrypts p as follows.

$$\text{Enc}(p, t) = \overleftarrow{F}\left(\left(\oplus_{j=1}^{m} K_j^{(p)_j}\right) \oplus e \oplus p \oplus W_t\right). \quad (9)$$

For duplicated tokens, the encryption can be simplified. The client record the ciphertext of the last occurrence of p, namely, Dup_token(p).

$$\text{Dup\_token}(p) = \left(\oplus_{j=1}^{m} K_j^{(p)_j}\right) \oplus e \oplus p \oplus W_t. \quad (10)$$

And the times of occurrences of p so far, namely, $c(p)$. For a token with content p, the client encrypts the token as follows.

$$\text{Enc}(p, c(p)) = \overleftarrow{F}\left(\text{Dup\_token}(p) \oplus W_{c(p)}\right). \quad (11)$$

Then the client computes Dup_token(p) to Enc(p, $c(p)$) and increases $c(p)$ by 1 (see Algorithms 2 and 3).

3.6. Token Inspection. To search for the occurrence of rules in packets, MB matches the encryption ruleset against the encrypted token sequence. To accord with the token encryption for duplicated tokens, MB initializes each Dup_token(R_i) to RE_i. When an encrypted token arrives, MB compares the token with each RE_i. If the token matches RE_i, MB updates Dup_token(R_i) in the same way as token encryption as follows.

Input: Token series, K, W
Output: Encrypt token series **for** each token p **do**
  if p occurs at the first time **then**
    $c(p) \leftarrow 0$
    $\text{Dup\_token}(p) \overleftarrow{\leftarrow} (\oplus_{j=1}^{m} K_j^{p_j}) \oplus e \oplus p$
    $\text{Enc}(p, c(p)) \leftarrow F(\text{Dup\_token}(p) \oplus W_{c(p)})$
  else
    $\text{Dup\_token}(p) \overleftarrow{\leftarrow} \text{Dup\_token}(p) \oplus W_{c(p)}$
    $\text{Enc}(p, c(p)) \leftarrow F(\text{Dup\_token}(p))$
  end if
  $c(p) \leftarrow c(p) + 1$
  output$(\text{Enc}(p, c(p)))$
end for

Algorithm 2: Token encryption.

$$\begin{aligned} \text{Dup\_token}(R_i) &= \left(\oplus_{j=1}^{m} KE_j^{(R_i)_j}\right) \oplus R_i \oplus W_0. \\ RE_i &= \overleftarrow{F}\left(\text{Dup\_token}(R_i) \oplus W_{c(R_i)}\right). \end{aligned} \tag{12}$$

The counter of occurrences of the R_i, namely, $c(R_i)$, is increased by 1. MB also takes actions such as disconnecting or issuing a warning to the user or administrator. The whole process is shown in Algorithm 3.

3.7. Token Validation. The receiver S runs the same tokenization and token encryption on the decrypted TLS/SSL traffic. S checks whether the plaintext of traffic is the same as the plaintext of the encrypted token sequence received from MB. Any inconsistency implies that C is malicious.

3.8. Detecting the Malicious MB. This section considers a stronger adversary MB that does not follow the protocol and applies chosen plaintext attacks to the proposed system. We present a mechanism for the client to detect the middlebox that uses faked rules. For a given MB, the RG generates some rule verification information that the clients can use to verify the honesty of the middlebox, that is, the middlebox uses the unfeigned rules in the rule encryption phase. First, RG determines all the parameters of the OTs between the given MB and clients. X is a 2-D array of integers, Y^0 and Y^1 are 1-D arrays of integers, where $X_{i,j}$, Y_j^0, and Y_j^1 are the ζ, y_0, and y_1 parameters of the OT for encrypting bit $(R_i)_j$, respectively. The first message of the OT between MB and C for $(R_i)_j$ can be expressed as follows:

$$o_{i,j} = \text{Enc}_{pk_c}\left(X_{i,j}\right) \oplus \left((R_i)_j \cdot Y_j^1 \oplus \overline{(R_i)_j} \cdot Y_j^0 \right). \tag{13}$$

The rule verification for R_i is defined as follows:

$$v_i = \text{Hash}\left(o_{i,1} \| o_{i,2} \| \cdots \| o_{i,m}\right). \tag{14}$$

The RG can compute each v_i as RG has all the parameters for computing each v_i. Then, RG sends each v_i to the client. In the followed rule encryption phase between MB and C, the client computes v_i' using the message in OT with the same method in equation (15). If $v_i = v_i'$, the client is sure that MB uses the unfeigned rule.

The above mechanism imposes a heavy burden on RG since RG must compute the verification for all the sessions between any clients and servers. An improvement is to use the garbled circuit. RG builds a garbled circuit to compute each v_i for a given middlebox. As all the parameters to compute v_i except for the client's public key are known before a session, the circuit's input is the client's public key. When a client starts a session, it asks for the garbled circuit for its corresponding MB from RG and computes each v_i with its public key as input. In rule encryption, the client computes the v_i from both the garbled circuit and messages from the MB and checks whether they are the same. In this scheme, RG only needs to build the garbled circuit for each MB rather than compute verifications for every pair of C/S.

4. Security and Correctness

4.1. Security and Correctness Requirements. The security definition follows BlindBox, PrivDPI, and Song et al. [27]. Either of the two endpoints in our system may be malicious, but at least one of the two endpoints is honest. This requirement is also essential for any intrusion detection system [28]. Because when both ends are malicious, they can collude to treat the middlebox.

There are some requirements implemented in methods such as BlindBox and PrivDPI. (A) MB can perform rule detection. MB can identify the substring of the payload matching a rule. (B) C/S cannot obtain the rules used by MB. This requirement prevents C/S from eluding detection. It also makes the rules-suppliers (RG) keep the confidentiality of the rules that are their pivot assets.

In addition to the above two requirements, we also achieve the same security requirements of BlindBox and PrivDPI: (C) MB cannot decrypt the payloads. (D-i) MB cannot decrypt the encrypted token sequence. (D-ii) MB cannot analyze the frequency of plaintext occurrence by the sequence of encrypted tokens. MB can only learn the number of occurrences of the rules in the session. MB cannot learn the frequencies of other tokens.

4.2. Correctness. We prove that the requirement (A) is met.

4.2.1. Correctness Definition. Correctness is defined as follows. (i) Assume that a substring s of the plaintext equals rule R_i. The MB will identify the corresponding token and report a matching of R_i. (ii) When s does not equal any rule, the probability of MB reporting a matching is negligibly small.

4.2.2. Correctness Guarantees. We first prove that MB encrypts the rule set correctly, that is, $RE_i = \text{Enc}(R_i, 0)$. According to equation (8), we have the following:

Input: Encrypted token series, RE, Dup_token, and W
Output: Whether the encrypted token is the same as the detection rule **for** $i = 1$ to n **do**
  $RE_i \leftarrow \overleftarrow{F}(\text{Dup\_token}(R_i) \oplus W_0)$
  $c(R_i) \leftarrow 1$
end for
for each encrypted token $\text{Enc}(p, t)$ **do**
  if $\text{Enc}(p, t) = RE_i$ **then**
    report a matching of R_i
    $\text{Dup\_token}(R_i) \leftarrow \text{Dup\_token}(R_i) \oplus W_{c(R_i)}$
    $c(R_i) \leftarrow c(R_i) + 1$
  end if
end for

Algorithm 3: Token inspection.

$$RK_i = \oplus_{j=1}^{m} KE_{i,j}^{(R_i)_j} = \oplus_{j=1}^{m} K_j^{(R_i)_j} \oplus KM_{i,j}. \tag{15}$$

As $e = \oplus_{j=1}^{m} KM_{i,j}$, we have

$$RK_i = e \oplus_{j=1}^{m} K_j^{(R_i)_j}. \tag{16}$$

Therefore,

$$RE_i = \overleftarrow{F}(RK_i \oplus R_i) = \overleftarrow{F}\left(\left(\oplus_{j=1}^{m} K_j^{(R_i)_j}\right) \oplus e \oplus R_i \oplus W_0\right) = \text{Enc}(R_i, 0). \tag{17}$$

Let the content of a token be R_i, and it is the first occurrence of R_i in the token series. The encrypted token is $\text{Enc}(R_i, 0)$. MB will detect the match as $RE_i = \text{Enc}(R_i, 0)$. Assume that a token with the content R_i is the t-th occurrence of R_i in the token series. The encrypted token is $\text{Enc}(R_i, t)$. Assume that on the MB side, we have $RE_i = \overleftarrow{F}(\text{Dup\_token}(R_i) \oplus W_t) = \text{Enc}(R_i, t)$ and MB detects the match. Then, for R_i's $(t+1)$-th occurrence, the encrypted token is $\text{Enc}(R_i, t+1)$. On the MB side, we have $RE_i = \overleftarrow{F}(\text{Dup\_token}(R_i) \oplus W_{t+1}) = \text{Enc}(R_i, t+1)$ and MB detects the match.

Therefore, the correctness definition is held. As keys for encryption are random, the ciphertexts of two different tokens may be the same, which we call the case a collision. The probability that a token and a rule leads to a collision is $1/2^m$ (the first type of birthday attack). For $m = 64$ or 128, this probability is $1/2^{64}$ or $1/2^{128}$. The correctness definition (ii) is held.

4.3. Security. We first show that when one of C and S is not honest, the honest MB can detect the case, and the honest S can detect the dishonest C. In the rule encryption stage, MB works out encrypted rules along with both C and S. If the two results are not the same, MB knows that one of C or S is not honest. S holds the session key and decrypts the SSL/TLS traffic. S can verify whether C sends the tokens in accord with the SSL/TLS traffic.

We prove that requirements (B), (C), (D-i), and (D-ii) are met. Because MB does not have the session key sk to decrypt the payloads and the threat requirement (C) has been met.

For requirement (B), According to the oblivious transfer, C cannot know the bit of each rule. We also hide the length of the rules. Requirement (B) is met.

We consider requirement (D-i) and requirement (D-ii). Since $\text{Enc}(p, t)$ is a one-way function and ρ, σ are not known by MB, MB cannot recover the plaintexts of encrypted tokens. The security definition (D-i) is held. For bit $b = (R_i)_j$, MB works out $K_j^b \oplus KM_{i,j}$ and knows nothing about $K_j^{1-b} \oplus KM_{i,j}$ by the OT. Given $(R_{i'})_j = b$, as $KM_{i,j}$ and $KM_{i',j}$ are different random bit strings, MB cannot recover K_j^b from KE_i and $KE_{i'}$.

As the same tokens are masked with different entries of W, the resulting encrypted tokens are different, which avoids frequency-based attacks from MB. Likewise, an eavesdropping adversary cannot recover the plaintexts of encrypted tokens and their frequencies. As OTs between C/S and MB protect the rules from leaking, the eavesdropping adversary cannot detect the matching of a rule. The security definition (D-ii) is held.

In this approach, we have fulfilled the requirement (D).

5. Performance Evaluation

We use OpenSSL-1.1.1a to implement encryption and message sending and Cppjieba to implement natural-language-processing-based tokenization. The one-way function we used is the Rabin function. We employ RawCap-0.2.0 to monitor the traffic and Wireshark-3.4.5 to collect statistics on the traffic. Each test is conducted 1,000 times, and the running time is the average time of the runs. The experiments use open-source rule sets and real and random network traffic. The detection rules are randomly inserted into test packets to measure the accuracy of MB by checking whether the rules are matched correctly.e conducted experiments on a PC with Intel(R) Core(TM) i5-6300U CPU with four cores at 2.20 GHz running 64-bit Windows 10.We use OpenSSL-1.1.1a to implement encryption and message sending and Cppjieba to implement natural-language-processing-based tokenization. The one-way function we used is the

Rabin function. We employ RawCap-0.2.0 to monitor the traffic and Wireshark-3.4.5 to collect statistics on the traffic. Each test is conducted 1,000 times, and the running time is the average time of the runs. The experiments use open-source rule sets and real and random network traffic. The detection rules are randomly inserted into test packets to measure the accuracy of MB by checking whether the rules are matched correctly.

We compare OTEPI with BlindBox and PrivDPI. The three approaches have the same security and threat model and conduct the same function. The existing machine learning and deep learning approaches only inspect the plaintext part of the traffic and perform different security functions from OTEPI.

5.1. Client (or Server). The client/server's main computation and communication overhead are in the token encryption step.

5.1.1. Tokens with Distinct Content. BlindBox performs AES encryption twice for one token. OTEPI performs m bitwise XORs and a one-way function on m-bit strings. For PrivDPI, one exponentiation operation, one multiplication operation, and one AES encryption operation are required. The comparison of the token encryption time is shown in Figure 4. It can be seen that the token encryption time is linear in the number of tokens in the three approaches. For the same amount of tokens, OTEPI consumes twice as much time as BlindBox and PrivDPI consumes 5.6 times as much time as BlindBox.

The token encryption is much faster than the rule encryption in both OTEPI and BlindBox. In the latter, BlindBox needs to transmit garbled circuits and encrypt and OTEPI needs to use OTs to transfer keys. However, in the latter, all these operations are not needed.

The NLP-based tokenization is more flexible than fix-length tokenization used in BlindBox and PrivDPI. For example, for the payload "login.html?usern-ame = bob," NLP can yield "login" and "username = bob" instead of "login.ht." Most NLP tools support dictionary-based segmentation, which is suitable for texts. The number of tokens is greatly reduced by discarding meaningless words such as "a" and "the" NLP-based tokenization transfers some computation of the MB to the client-side. An MB is usually heavy-loaded, and it is desirable for the client to share the load.

The sliding window based tokenization leaks information about the payload length. Tokenization by NLP can hide the payload length since meaningless words are not recorded.

The running time of the tokenization by NLP is 2.3 to 2.8 times more than that of the sliding window. The time used for NLP tokenization is shown in Figure 5.

As NLP reduces the number of tokens, the time of token encryption and matching are reduced. In Figure 6, we compare the time of tokenization and token encryption of the client in BlindBox, PrivDPI, and OTEPI. It shows that by using NLP tokenization, OTEPI becomes the fastest in token encryption.

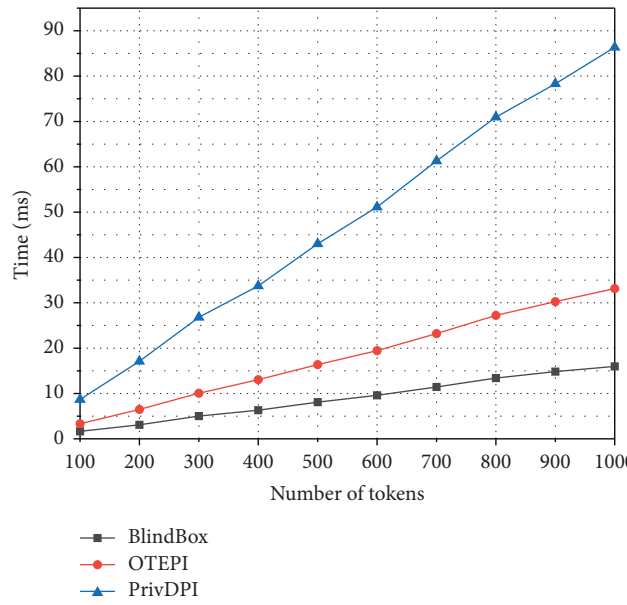

Figure 4: Encrypt tokens in client.

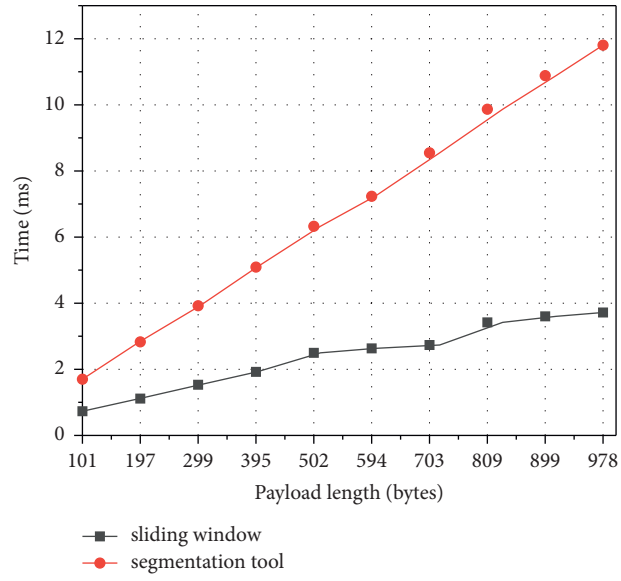

Figure 5: Tokenization time.

5.1.2. Tokens with Duplicate Content. When tokens repeat, the encryption time is different from that with unique content. The client (or server) uses the recorded encrypted tokens for acceleration in all three approaches. The encryption method for an existing token p in BlindBox is $\mathrm{AES}_{\mathrm{AES}_{\mathrm{ckey}}(p)}(\mathrm{salt} + t)$, where t is the number of occurrences of p. The re-encrypting method in OTEPI can be found in equation (12). PrivDPI uses table lookup to compute the exponentiation and multiplication operations for duplicated tokens, such that only one AES operation is needed.

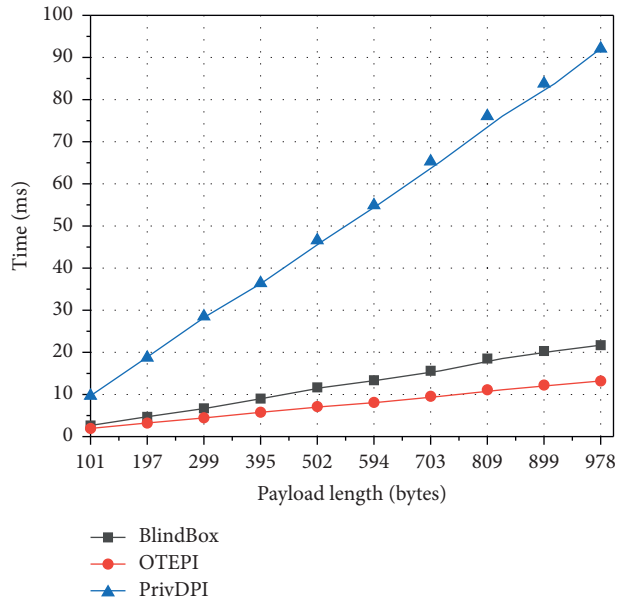

Figure 6: Tokenization and encryption of tokens.

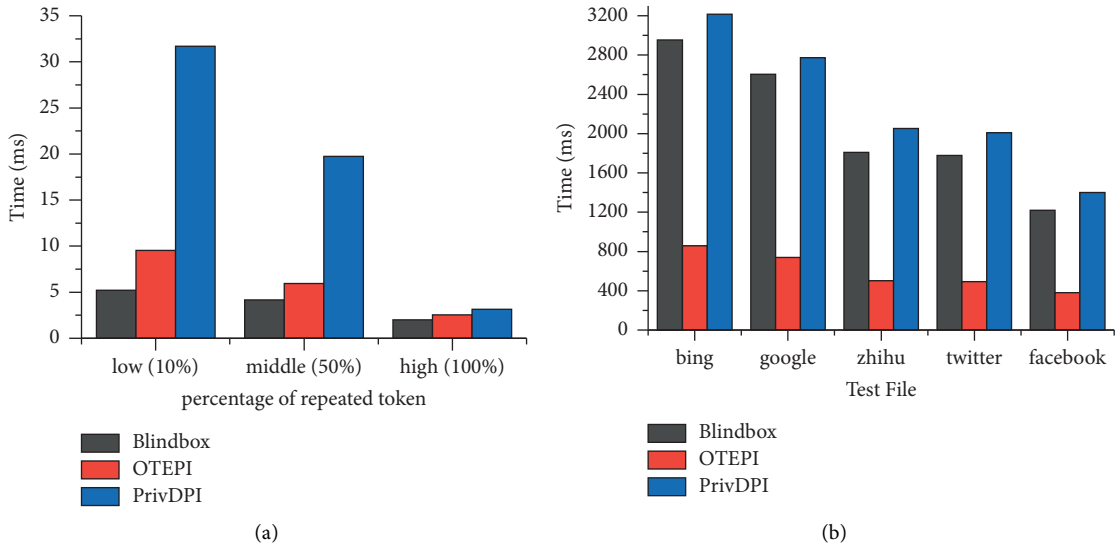

Figure 7: Token encryption time.

We evaluate the encryption time for the traffic with different percentages of repeated tokens. Repeated tokens are common in the real world. For example, when searching for recipes and travel brochures online, multiple queries will return similar results.

In Figure 7(a), We use 500 tokens, among which 10% to 100% tokens are repeated tokens. When the repetition rate of the token is 100%, the encryption time of OTEPI and BlindBox are the same.

In Figure 7(b), we show the computational overhead of the server in encrypting an HTML web page accessed the second time. In Figure 7(a), for the recurring token, our encryption is faster than BlindBox and PrivDPI. The running time of BlindBox is about 3.5x of OTEPI, and PrivDPI is about 3.8x of OTEPI.

5.2. Middlebox

5.2.1. First Session. For MB, the time required for encrypting rules and the communication overhead for obtaining these encryption rules are shown in Table 3.

The high bandwidth consumption of BlindBox is due to the garbled circuits. In OTEPI, bandwidth consumption is significantly reduced compared to BlindBox, for the low bandwidth consumption of OT in the rule setup. PrivDPI only needs to send a few group elements per rule, which only incurs very low bandwidth.

A comparison of the rule encryption time of the three approaches is shown in Table 4. OTEPI has a high time consumption because each rule requires m times OTs(64 or 128). BlindBox requires one garbled circuit per rule,

Table 3: MB: bandwidth (first session).

Number of rules (8 bytes)	Bandwidth		
	BlindBox	OTEPI	PrivDPI
1	16.72 MB	28.88 KB	57.16 B
3000	50.16 GB [13]	82.51 MB	172.83 KB

Table 4: MB: time (first session).

Number of rules (8 bytes)	Time		
	BlindBox (s)	OTEPI (s)	PrivDPI (s)
1	0.956	0.928	0.223
3000	294.495	267.249	1.021

Table 5: Session bandwidth.

Number of sessions	BlindBox (GB)	OTEPI (MB)	PrivDPI
1	50.16	82.51	49 B
5	250.845	82.526	0.291 MB
10	501.69	82.542	0.292 MB
20	1003.38	82.558	0.293 MB [13]

while PrivDPI only requires one exponentiation. In BlindBox, the communication transmission between MB and C/S is a garbled circuit of the function $F = \mathrm{AES}_{\mathrm{ckey}}(\cdot)$, where *ckey* is the key of the client-side to encrypt tokens. Using the garbled F, MB encrypts rule R_i. Then BlindBox adds a random number salt and computes $\mathrm{AES}_{\mathrm{AES}_{\mathrm{ckey}}(R_i)}(\mathrm{salt} + t)$ as the ciphertext of rule R_i in its t-th occurrence. In OTEPI, the computation costs mainly come from the oblivious transfer. In both BlindBox and OTEPI, the setup of encrypted rules has a high cost, so there is a huge gap in time.

5.2.2. Subsequent Sessions. We compare the bandwidth usages between OTEPI and BlindBox in case of multi-sessions. We still use 3000 rules in the tests. The results are shown in Table 5.

In subsequent sessions, OTEPI consumes less bandwidth than BlindBox. BlindBox needs to generate a garbled circuit for each rule in each session. PrivDPI transmits rules encryption parameters in the first session. PrivDPI can reuse the obfuscated rules set up in the first session in subsequent sessions, and only sends one group element in each followed session.

In terms of bandwidth consumption of multiple sessions, though not as efficient as PrivDPI, OTEPI significantly reduces the bandwidth consumption of establishing encryption rules compared with BlindBox. Meanwhile, OTEPI also achieves the reusable obfuscated rule as PrivDPI.

5.2.3. Accuracy of Tokenization. The accuracy of tokenization impacts the recognition accuracy of the system. OTEPI, BlindBox, and PrivDPI detect the matching when the token matches a rule. In this set of experiments, rules from different rule sets are randomly inserted into the traffic, and the accuracy of *MBs* using different tokenization methods in matching is tested.

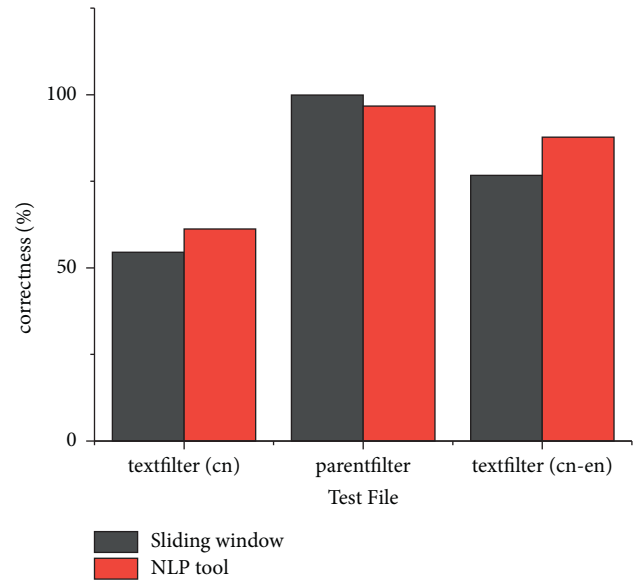

Figure 8: Accuracy comparation.

We use three rule sets in the accuracy test. The testfilter(cn) [29] is a pure Chinese ruleset, and parentfilter [30] is a parent filter ruleset, and testfilter(cn-en) [31] is a ruleset mixed with Chinese and English rules. As shown in Figure 8, for the parents-filtering rules, BlindBox has a higher accuracy rate than OTEPI. This is because parents-filtering rules are long, and NLP tools divide a rule into several words, e.g., "zippyvideos" is divided into "zippy" and "videos," which affects the accuracy. For testfilter(cn) and testfilter(cn-en), each Chinese character occupies 2-3 bytes under UTF-8 encoding. In BlindBox, rules shorter than the sliding window may be missed. As an example, the first token of text "adult check" is "adult ch" under 8 byte window. The rule "adult" will be missed. The fixed-length tokenization is not as accurate as NLP tokenization because sensitive words are always short. The famous anonymous website 4chan, for example, does not have any board with a name longer than four characters and uses the shortened form of multisyllabic words.

5.3. Summary. Compared with BlindBox, we significantly reduced the communication bandwidth from 50 GB to 82 MB in the rule encryption. Although our bandwidth consumption is higher than PrivDPI, the rule encryption is faster than PrivDPI. Without NLP tokenization, our token encryption is 2.6 times faster than PrivDPI. When NLP tokenization is used for HTML or other plaintext data, OTEPI achieves 1.7x speedup on BlindBox and 7.6x speedup on PrivDPI. In terms of accuracy, OTEPI has a higher recognition rate than BlindBox and PrivDPI for short rules and a slightly lower recognition rate than BlindBox for parent-filtered URL rules.

6. Discussion

Many directions can be developed in the future under the scheme proposed in this paper. Advances in NLP technology that produce fewer, more accurate, and fewer tokens can improve the accuracy and computational performance of OTEPI. The bandwidth overhead of oblivious transfer is still more significant than that of PrivDPI when encrypting rules. Finding or optimizing an oblivious transfer algorithm that saves more communication traffic can bring better bandwidth performance to OTEPI. OTEPI currently supports middleboxes for DPI filtering only. The machine learning approaches can also benefit from the multi-party security computing. We believe that the general blueprint OTEPI provides can extend the machine learning approach to process the encrypted payloads.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Commission of Development and Reform of Jilin Province No. 2019C053-10 and the Education Department of Jilin Province No. JJKH20190162KJ.

References

[1] M. Meeker, "Internet trends 2019," Tech. rep, Bond, San Francisco, 2019.

[2] Google, "Google transparency report," Google, California, Tech. rep, 2020.

[3] S. Ruoti, J. Andersen, T. Monson, D. Zappala, and K. Seamons, "A comparative usability study of key management in secure email," in *Proceedings of the 14th Symposium on Usable Privacy and Security (SOUPS 2018)*, pp. 375–394, USENIX Association, Berkeley, California, August, 2018.

[4] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox: deep packet inspection over encrypted traffic," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 213–226, Association for Computing Machinery, New York, NY, USA, August 2015.

[5] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Communications of the ACM*, vol. 28, no. 6, pp. 637–647, 1985.

[6] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pp. 535–548, Association for Computing Machinery, New York, NY, USA, November 2013.

[7] M. Naor and B. Pinkas, "Oblivious transfer with adaptive queries," in *Proceedings of the 19th Annual International Cryptology Conference*, pp. 573–590, Springer-Verlag, Berlin, August 1999.

[8] Y. Lindell and B. Pinkas, "A proof of security of yao's protocol for two-party computation," *Journal of Cryptology*, vol. 22, no. 2, pp. 161–188, 2009.

[9] A. C. C. Yao, "How to generate and exchange secrets," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (SFCS 1986)*, pp. 162–167, IEEE, Manhattan, New York, June, 1986.

[10] D. Naylor, R. Li, C. Gkantsidis, T. Karagiannis, and P. Steenkiste, "And then there were more: secure communication for more than two parties," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pp. 88–100, New York, NY, USA, December 2017.

[11] S. Canard, A. Diop, N. Kheir, M. Paindavoine, and M. Sabt, "BlindIDS: market-compliant and privacy-friendly intrusion detection system over encrypted traffic," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 561–574, New York, NY, USA, April 2017.

[12] 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*, vol. 25, no. 6, pp. 3753–3766, 2017.

[13] J. Ning, G. S. Poh, J. C. Loh, J. Chia, and E. C. Chang, "Privdpi: privacy-preserving encrypted traffic inspection with reusable obfuscated rules," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1657–1670, New York, NY, USA, November 2019.

[14] A. Yamada, Y. Miyake, K. Takemori, A. Studer, and A. Perrig, "Intrusion detection for encrypted web accesses,"vol. 1, pp. 569–576, in *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, vol. 1, IEEE, Manhattan, New York, may 2007.

[15] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, pp. 35–46, Association for Computing Machinery, New York, NY, USA, October 2016.

[16] B. Anderson, S. Paul, and D. McGrew, "Deciphering malware's use of tls (without decryption)," *J. Comput. Virol. Hacking Tech.*vol. 14, no. 3, pp. 195–211, 2018.

[17] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, Article ID 102419, 2020.

[18] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, and A. Pescapé, "A dive into the dark web: hierarchical traffic classification of anonymity tools," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1043–1054, 2019.

[19] K. L. K. Sudheera, D. M. Divakaran, R. P. Singh, and M. Gurusamy, "Adept: detection and identification of correlated attack stages in iot networks," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6591–6607, 2021.

[20] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: a flow sequence network for encrypted traffic classification," in *Proceedings of the IEEE INFOCOM 2019-IEEE Conference On Computer Communications*, pp. 1171–1179, IEEE, Paris, France, May, 2019.

[21] K. Cho, B. Van Merriënboer, C. Gulcehre et al., "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014, https://arxiv.org/abs/1406.1078.

[22] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Distiller: encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183, Article ID 102985, 2021.

[23] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proceedings of the 2017 IEEE international conference on intelligence and security informatics (ISI)*, pp. 43–48, IEEE, Beijing, China, July, 2017.

[24] Y. C. Chen, Y. J. Li, A. Tseng, and T. Lin, "Deep learning for malicious flow detection," in *Proceedings of the 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–7, IEEE, Montreal, QC, Canada, October, 2017.

[25] D. Goltzsche, S. Rüsch, M. Nieke et al., "Endbox: scalable middlebox functions using client-side trusted execution," in *Proceedings of the 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 386–397, IEEE, Manhattan, New York, June 2018.

[26] J. Kilian, "Founding crytpography on oblivious transfer," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pp. 20–31, New York, USA, may 1988.

[27] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pp. 44–55, IEEE, Manhattan, New York, may 2000.

[28] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[29] azure, "Chinese banned words," Gitee, Tech. rep, 2016.

[30] F. Prigent, "Parents filter url," Université Toulouse 1, Capitole, Tech. rep, 2022.

[31] wear, "Chinese and English banned words," Tech. rep, 2015.