Electronic Preprint for Journal of Information Processing Vol.31

Paper on Consumer Systems

Service Identification of TLS Flows Based on Handshake Analysis

Ryo Asaoka[1,a)] Yuto Soma[1] Hiroaki Yamauchi[1] Akihiro Nakao[2] Masato Oguchi[3]
Saneyasu Yamaguchi[1] Aki Kobayashi[1]

Received: January 22, 2022, Accepted: November 14, 2022

Abstract: Identification of services constituting traffic from given IP network flows is important for many purposes such as management of quality of service, prevention of security problems, and providing a discounting service for customers only in accessing specified services like zero-rating service. The simplest methods for identifying these services are identifications based on IP addresses and port numbers. However, such methods are not sufficiently accurate and thus require improvement. Deep packet inspection (DPI) is an advanced method for improving the accuracy of identification. Many current IP flows are encrypted with the transport layer security (TLS) protocol. Therefore, an identification method cannot analyze almost all the data encrypted by TLS. In the cases of TLS 1.2 or less, some fields, e.g. server name indication (SNI), in the protocol header for the TLS session establishment are not encrypted and then can be analyzed. Thus, we can expect that the service can be identified from IP flows, which are composed of TLS sessions, by analyzing these fields. For achieving this, two challenges are mainly required. One is grouping TLS sessions by accesses from many TLS sessions that pass through a network element. The other is the identification of service from TLS sessions grouped in the first challenge. In our work, we mainly focus on the second theme, i.e., service identification from given TLS sessions. In our previous work, we proposed a method for identification by analyzing these non-encrypted data based on DPI and n-gram. However, there is room for improvement in identification accuracy because this method analyzed all the non-encrypted data including random values without protocol analysis. In this paper, we propose a new method for identifying the service from given TLS sessions based on SNI with protocol data unit (PDU) analysis. The proposed method clusters TLS sessions according to the value of SNI and identifies services from the occurrences of all groups. We evaluated the proposed method by identifying services on Google, Yahoo, and MSN sites, and the results showed that the proposed method could identify services more accurately than the existing method. The average ratios of inaccurate identifications were decreased by 65%, 72%, and 41% in our experiments of Google, Yahoo, and MSN services, respectively.

Keywords: service identification, TLS, SNI, HTTPS

1. Introduction

The World Wide Web provides a variety of services such as web page search and video streaming. Service identification from given Internet flows in a network element is important for various purposes, for example, the management of quality of service (QoS), prevention of security problems, and providing a new service that allows a customer to access a specified service without network usage charge, which is usually called zero-rating service. The most basic identification methods rely on IP addresses and port numbers [1]. However, these basic methods cannot provide high accuracy of identification. In some cases, many services are provided by a site with only one IP address. In some cases, a dynamic port control [2] is used. Thus, the accuracy of identification based only on such information is limited. Inspection of packet payloads, called deep packet inspection (DPI), is an advanced method for solving this problem. Previous studies [3], [4] demonstrated that DPI improved the accuracy. However, many recent communication flows are encrypted [5] with the transport layer security (TLS) protocol. Identification methods cannot analyze almost all the packet payloads that are encrypted by TLS. Some fields in TLS headers are not encrypted and can be analyzed. For example, the server name indication (SNI) field is not encrypted in TLS 1.2 or less. We then expect that the service can be identified based on the information in these non-encrypted fields. This service identification is achieved mainly by two challenges. The first challenge is clustering TLS sessions into accesses that they belong to. The second challenge is identifying a service from given TLS sessions that are clustered in the first challenge. We focus on the second challenge. The first challenge can be nearly or partially achieved by some existing works such as a method for application identification [6], [7]. This method groups TLS sessions by the applications. In some cases, such as situations wherein a web browser is holding only one tab for opening a web page or the browser does not download plural websites simultaneously, the application identification by this method achieves the TLS session clustering.

In this paper, we discuss a method for identifying the service

[1] Kogakuini University, Shinjuku, Tokyo 163–8677, Japan
[2] The University of Tokyo, Bunkyo, Tokyo 113–8654, Japan
[3] Ochanomizu University, Bunkyo, Tokyo 112–8610, Japan
[a)] cm22002@ns.kogakuin.ac.jp

from given sessions encrypted by TLS 1.2 or less by inspecting packet payloads. There are two types of service identifications, which are identifications from limited candidates and those from unlimited services. We studied the former identifications. Namely, an identification method chooses the most likely service from the preliminary specified candidates. We defined service as a function provided by a site, which is almost the same as the service of an application service provider (ASP). For example, Gmail, Google Web search, and YouTube are different services. A portal site that contains various services, e.g., yahoo.com, is not a service but a connected site. A usual user opens a Web page via a browser by inputting a URL string. We define this opening as "one access." One access causes multiple connections. In the case of HTTPS (Hypertext Transfer Protocol Secure), each connection contains a TLS session.

We propose two identification methods in this paper. The first method identifies the service based on the value in the SNI field in the first TLS session. The second method clusters TLS sessions according to the value of the SNI field, and the service was identified based on the groups contained in the TLS sessions in one access. In the cases of TLS 1.2 or less, SNI is a field in the *ClientHello* message, which is not encrypted. In the cases of TLS 1.3, SNI is encrypted. Thus, our proposed methods are for TLS 1.2 or less. This paper is based on an earlier conference work-in-progress paper [3] and a conference workshop short paper of Ref. [4].

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes the proposed methods. Section 4 evaluates the proposed methods. Section 5 presents discussions. Finally, Section 6 concludes this paper.

2. Related Work

2.1 N-gram

N-gram is a contiguous sequence of n items in a document, which is typically used in the text search field and can be applied to text data or binary data.

For example, in the case of 2-gram for text data in **Fig. 1**. "To" is an instance of 2-gram. It appears at the locations of the first letter and the 14th letter. "O" is also an instance of 2-gram. It appears at the locations of the second letter and the 15th letter.

Next, we explain byte-based n-gram for a byte sequence. In the case of the binary data in Fig. 1, "0xF4-0xEF" is an instance of byte-based 2-gram. It appears at the locations of the first byte and the 11th byte. "0xEF-0xA0" is also an instance of byte-based 2-gram. We can find this sequence at the locations of the second, eighth, 12th, and 15th bytes. In this paper, we call this n-gram for binary data byte-based n-gram.

2.2 Service and Application Identification

In this subsection, we introduce the existing service and application identification methods. In many cases, service types and port numbers are closely related [1]. An identification based on the IP address or the port number is the most basic and easiest method of identification. However, this method has low accuracy and there is room for improvement [6]. In some cases, e.g., when using network address translation (NAT) or reverse proxy, this

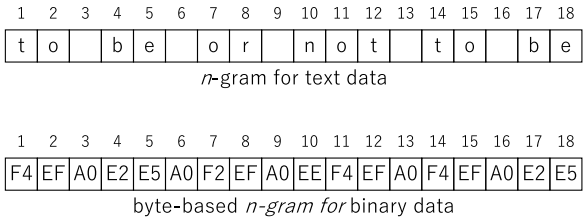


Fig. 1 1 n-gram.

method does not work well [2]. In some cases, a site may provide several services from the same IP address, which changes dynamically. Furthermore, many services are provided via the Web through port number 80 or 443, then a service is hardly identified from its port number. The port number or the IP address is effective for identifying the accessing protocol and connected site, but these are not useful for identifying services. Gigis et al. analyzed certificates in TLS in the Internet traffic and showed huge content providers' server deployments [5]. Their work does not contribute to service identification, but demonstrated that investigation of some non-encrypted parts of TLS sessions presented important findings.

Velan et al. [8] reviewed previous studies on information extraction from encrypted traffic. All network protocols can ensure secure data transfer with encryption using a non-encrypted initialization phase, and these data can easily be extracted and used for monitoring network traffic [8]. Qualys, Inc. has developed a method to obtain the fingerprints of a client based on the SSL (Secure Socket Layer)/TLS initial handshake [9]. The p0f tool [10] extracts information on peers' systems, such as the operating system and user agents, by monitoring the traffic. Holz et al. [11] presented a comprehensive analysis of X.509 certificates in traffic, and revealed the quality of certification. Husák et al. [12] proposed real-time lightweight client identification based on network monitoring and TLS fingerprinting. In the site of Ref. [9], a method for utilizing Handshake in SSL as a fingerprint of HTTP Client is presented. Iwai et al. [13] proposed an application identification method using machine learning, which was capable of achieving 82% accuracy. However, the method required five days to complete the learning stage. Hara et al. [14] proposed a connected site identification method, which analyzed packet payloads and created an n-gram database. These studies did not consider identification based on multiple connections; thus, these cannot successfully identify a service containing multiple connections. Moreover, these studies identify their clients, applications, or connected sites rather than the services from traffic. Therefore, their methods did not directly contribute to service identification. Their objectives were different from our objectives.

Hara et al. [15] proposed a method for identifying a service from given flows, which achieved 100% accuracy on ten Google services. However, the method was not applied to other Google services or services of other ASP sites. In this paper, this existing method is referred to as the *n-gram method*.

Shbair et al. surveyed HTTPS traffic and services identification approaches [16]. They introduced a variety of methods for identification including protocol-structure-based methods using SNI.

Bortolameotti et al. detected malicious TLS connections by using both SNI and SSL certificate information [17]. They examined the Levenshtein distance between the SNI and the top 100 most visited websites, the structure of the server-name string in SNI, and the format of the server-name string. Shbair et al. focused on HTTPS traffic filtering based on SNI and evaluated the reliability of SNI for identifying and filtering HTTPS traffic [18]. They then showed a method for bypassing a firewall based on their investigation of SNI. However, their discussion covers only a single connection, unlike this paper. They did not discuss the use of multiple SNIs in multiple connections. Kim et al. proposed a method to classify network traffic by their application services [19]. They used the certificate publication information field in TLS Handshake, which was not encrypted. Similar to the other previous work and our work, this work used a non-encrypted part of TLS. They then showed that their method could identify services with their simple experiment. This work is unique because it used also session ID. However, they did not use occurrence vectors of a certificate, SNI, or other information that covered multiple connections. It was not shown that the method achieved high accuracy in more complicated evaluations like our evaluation.

There are some works on service identification based on other information. Ding et al. proposed an identification method of service based on machine learning (random forest) [20]. This method used different information from ours. The method used the features like packet length and interval time. Yang et al. proposed a method for the classification of TLS flows based on deep learning [21]. This method also requires training and is based on packet length and packet inter-arrival time. Shbair et al. proposed a method based on many features such as the number of packets, packet size, inter-arrival times, and their statistical values [22]. This method is also based on machine learning. Barut et al. proposed a method for application classification from encrypted TLS flows based on machine learning [23]. This method used the features like port numbers, the number of packets, the number of bytes, the time duration of the flow, etc. This method selected features before training its model with some methods such as a random forest. This method also is based on machine learning. Like the work of Ref. [13], these methods also required training in machine learning. Therefore, a method without training and machine learning, like our method, is also essential in addition ion to these methods. These methods were based on some information based on not service but network environment, for example, inter-packet arrival time. These methods' accuracies are expected to depend on other traffic, while our method does not depend on network conditions.

2.3 DPI

DPI is a packet filtering method at a network element. Its concept consists of the analysis of the contents of the captured packets to accurately and timely discriminate the traffic [24]. It analyzes the payload of a packet to detect spam, malicious software, or other attacks. A network element responsible for DPI collects statistical information and determines whether the packet passes based on the information.

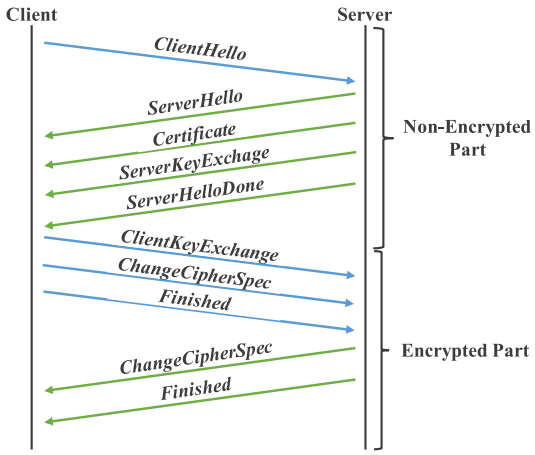


Fig. 2 TLS Handshake.

2.4 TLS Handshake

A TLS Handshake is a process to establish a TLS session. **Figure 2** illustrates the message sequence of this handshake. The *ClientHello* message from the client to the server describes lists of TLS protocol versions, cipher suites, compression algorithms that can be used by the client, and a random number used as input to the process for generating a key. The *ServerHello* message specifies the version of the TLS protocol, cipher suite, and compression algorithms to be used in the encrypted session. The *Certificate* message transmits the server certificate and the intermediate certificate. The *ServerKeyExchange* message specifies the public key used for encryption when transmitting the common key. The *ServerHelloDone* message indicates the completion of the transmission of the Handshake messages from the server. The client then verifies the server certificate. The *ClientKeyExchange* message specifies the common key encrypted with the public key transmitted from the server. Using the *ChangeCipherSpec* message and the *HandshakeFinished* message, both the client and the server confirm the changes in the cryptographic specification and the completion of the establishment of the TLS session, respectively.

The *ClientHello*, *ServerHello*, *Certificate*, *ServerKeyExchange*, and *ServerHelloDone* messages in Handshake messages are not encrypted in TLS 1.2 or less. Therefore, the features of services can be extracted by analyzing the payloads of these messages based on DPI.

2.5 Service Identification Method Based on N-gram

Here, we explain the n-gram method [15]. Hara et al. [15] proposed a service identification method based on n-gram.

Figure 3 shows the service flow model of the work of Ref. [15]. We also assume this model in this paper. This assumed that multiple TLS sessions were established for each access to a service. For instance, approximately 10 TLS sessions are established when a user accesses "www.google.com" using a Web browser, which is regarded as "one access" in this study. One TLS session is established in each TCP connection. The n-gram method clustered these sessions into several groups. In the case of this figure, five TLS sessions are established. Three of these are clustered into *Group A*, while the other two are clustered into *Group B*.

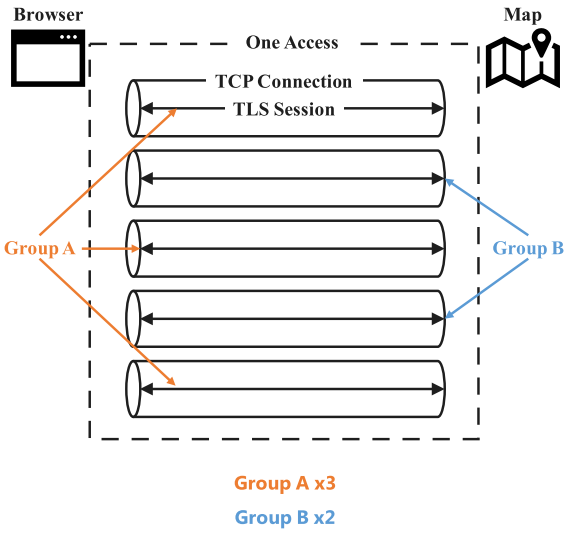


Fig. 3 Traffic Model of One Accesses to the Map Service.

Transmitted data after sending the common key are encrypted; thus, extracting features by inspecting them is almost impossible. The authors focused on packets sent from a server to a client because data from a client to a server mainly contained information on the client and might limit identification. Thus, the n-gram method analyzes the *ServerHello*, *Certificate*, *ServerKeyExchange*, and *ServerHelloDone* messages.

This method consists of two phases: the preliminary investigation phase and the identification phase. In the preliminary investigation phase, the following steps are executed.

(1) Access to each service is performed and their traffic is captured. The traffic is composed of TLS sessions.
(2) The non-encrypted parts in the handshake messages of every TLS session are analyzed, the frequencies of all the n-grams in the parts are counted, and an n-gram frequency database is created. The database stores the relationship between sessions and frequencies of n-grams in it.
(3) These sessions are clustered with the correlation coefficient between frequencies of n-grams of sessions.
(4) For each access, the number of sessions in each group is counted. This number is called the *group frequency*. The database that stores the relationship between the traffics captured in this preliminary investigation phase and the group frequencies is created. This is *group frequency database*.

In the paper [15], 2-gram was used as an instance of the n-gram. As the messages included not only text data but also binary data, the 2-gram is a byte-based n-gram. Step (2) counted the frequencies of all the byte-based 2-grams, which were 0x00-0x00, 0x00-0x01, 0x00-0x02, and so on. This method analyzes the data between the message for choosing a protocol and that for a sending server's certification.

In the identification phase, the following steps are executed.

(1) The traffic for identification is captured. It is composed of TLS sessions
(2) In each TLS session, the n-gram frequency of every byte-based n-gram is calculated, and the correlation coefficient between the frequencies of this session and every session in the database is calculated. This method then classifies this session into the group that has the largest average correlation coefficient.
(3) Based on the previous step, the number of sessions of every group is counted. This is the *group frequency* of the traffic for identification.
(4) The distance between the group frequencies of the traffic for identification and traffic in the group frequency database is calculated using the Manhattan distance. Then, the service with the smallest distance is determined as the identification result.

This method calculates a correlation coefficient between two vectors significantly many times. This method is severely time-consuming and was evaluated with a small number of flows in the paper [15]. We think that this method is not scalable and should be modified to be more lightweight.

2.6 Clustering TLS Sessions and Clustering Abilities of the Fields in TLS Handshake

Here, we introduce discussions on clustering the TLS sessions based on the non-encrypted parts in TLS Handshake.

An existing method [15] counted the frequency of every 2-gram in the non-encrypted part of the downloading flows and created a vector composed of the frequencies of all 2-grams. The method then calculated the correlation coefficient between the vectors of two sessions and clustered the sessions with a correlation coefficient higher than the threshold. This method did not analyze the byte stream according to the TLS protocol specification. This method searched also the field of random values for 2-grams. We expect that this behavior declined the accuracy of this discussion.

Since the work of Ref. [15] demonstrated that the sessions could be clustered using the values in the non-encrypted part, the work of Ref. [25] investigated the ability to cluster of each field in the TLS Handshake message in downloading flows, and the work of Ref. [26] investigated the ability of each field in the ClientHello message. The work then revealed that SNI, which was a field in the ClientHello message, could classify the sessions into a sufficient number of groups. However, the work did not present a discussion on service identification.

3. Service Identification Methods Based on SNI Analysis

In this section, we propose two methods for service identification based on the SNI analysis: identification based on the hostname in the SNI in the first TLS session [3] and identification based on SNI occurrence [4]. The latter method is a more advanced one and the former method is nearly a variant of the latter method. We propose the latter method in Section 3.1. We then propose the former method in Section 3.2 as a variant.

3.1 Identification Based on SNI Occurrence

The method based on SNI occurrence is composed of a preliminary investigation phase and an identification phase like the existing method [15]. In the preliminary investigation phase, this method creates a database for the relationship between the service and the occurring SNIs in one access for the service. *An access*

is defined in the second paragraph in Section 1. In the identification phase, this method chooses the most likely service from the given candidate services based on this database using Bayesian inference.

The preliminary investigation phase includes the following steps.

(1) This method accesses each candidate service multiple times and captures their traffic.
(2) This method checks the occurrence of each SNI value (whether the value is included in the SNI fields in all the TLS sessions) in each access. The set of occurrences is called the *occurrence vector*.
(3) The method creates a database of the relationship between each service and its occurrences, called the *SNI occurrence database*.

Figure 4 illustrates the preliminary investigation phase. In this case, this method accesses a mail service, and five TLS sessions are established. Each session has an SNI field, and three types of SNI values (sni1.com, sni2.com, and sni3.com) occur. Thus, the *occurrence vector* is (1, 1, 1, 0). This vector is registered in the *SNI occurrence database*.

The identification phase includes the following steps.

(1) This method captures the traffic to be identified.
(2) This method checks the occurrence of each SNI in all the TLS sessions in the traffic and creates its *occurrence vector*.
(3) The probability of each candidate service is calculated using Bayesian inference with Eq. (1). The service with the highest probability is considered the identification result.

$$P(A \mid X) = \frac{P(X \mid A)P(A)}{P(X)} \quad (1)$$

where $P(A \mid X)$ is the probability of service A in the condition that the *occurrence vector* X is observed. $P(A)$ is the probability that service A occurs. $P(X)$ is the probability that the *occurrence vector* X is observed, calculated from the *SNI occurrence database*. $P(X \mid A)$ is the probability that the *occurrence vector* X appears while accessing service A, calculated using the *SNI occurrence database*.

In practical usage and our evaluation, the true probabilities cannot be obtained. Therefore, these probabilities are obtained approximately from its limited observation or given in the experimental setup in the evaluation.

3.2 Identification Based on Hostname in SNI in the First Session

The identification method based on the hostname in the SNI in the first TLS session restricts the number of dimensions of the occurrence vector to be one and uses only the first TLS session in one access. The SNI field of the first TLS session in each access describes the hostname in the URL of the accessed service. This proposed method identifies the service by the hostname only if the hostname is unique in all the candidate services. If the hostname is not unique but shared with n services, the candidate services are narrowed down to n services, and the identification is completed. The service is not uniquely chosen, and "one of the n services" is the identification result. This method analyzes only the first session. Therefore, this method consumes little CPU time.

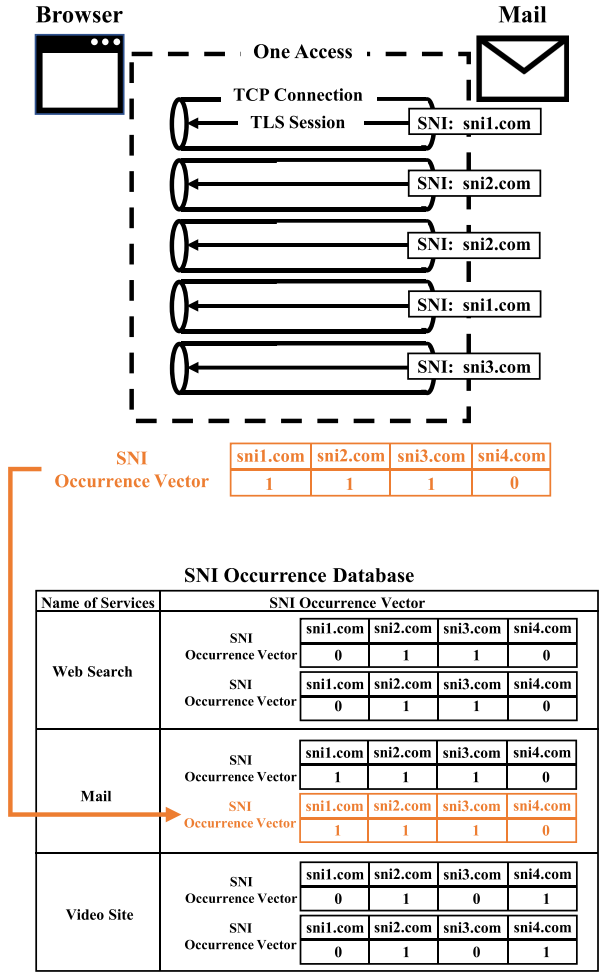


Fig. 4 SNI occurrence vector and database.

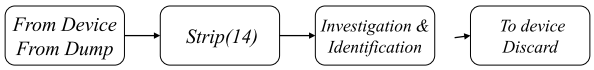


Fig. 5 Implementation of prototype.

3.3 Design and Prototype Implementation

First, we explain our prototype implementation of the proposed methods. We implemented our prototype system by Click Modular Router. **Figure 5** shows its configuration file, which is a kind of source file, of the elements of the Click Modular Router of our prototype implementation. It contains four elements. The first element provides packet data from a network interface card device or a dump file. The second element deletes the first 14 bytes of the packet for trimming the Ethernet header. The third element is a third-party Click element that we implemented. This element creates a database in the preliminary investigation phase and identifies its service with the Bayesian inference based on the database in the identification phase. The fourth element transmits the packet data to a network interface card device or discards them. For practical use, this implementation inputs packet data from a network interface card, determines the identification result, and outputs the packet data to another network interface card. For performance evaluation, this inputs packet data from a pre-recorded packet dump file, determines the identification re-

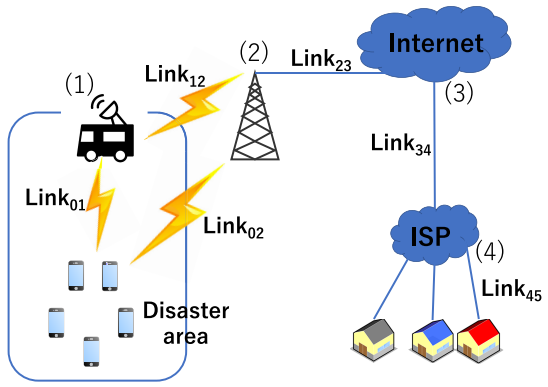


Fig. 6 Sample situation of usage.

sult, and discards the data.

Next, we describe a design and an assumed situation wherein a user utilizes the proposed method. **Figure 6** illustrates a sample situation in a heavy disaster. In the figure, the nodes (1), (2), (3), and (4) represent the network elements, i.e., network routers. Link_{01}, Link_{02}, Link_{12}, Link_{23}, and Link_{34} are the links between these routers and nodes. When a heavy disaster occurs, these links will be severely congested. In particular, Link_{01}, Link_{02}, Link_{12}, and Link_{23} will be congested. We expect that service identification and giving priority are performed as follows.

(1) at a mobile base station car near a struck area

Link_{01} and Link_{12} may be severely congested. The traffics of some important services should be provided higher priorities by identifying the service at the router in order to avoid being out of service due to too heavy congestion.

(2) at a usual base station

Link_{02} and Link_{12} may be congested and some important services' traffics should be given higher priority.

(3) at a core router

Even when Link_{01}, Link_{02}, and Link_{12} are congested, the total amount of traffic at a core router in the Internet is remarkably higher than those at base stations like (1) and (2). Thus, we expect that identification at (3) requires very high computing resources and is difficult. If it is possible, network slicing based on identified traffic is an effective way.

(4) at an edge router of ISP

The congestions at Link_{34} and Link_{45} will be less than those at Link_{01}, Link_{02}, and Link_{12}. In the cases Link_{34} and Link_{45} are also heavily congested, the prioritization by identification will be effective for transmitting important information.

4. Evaluation

This section presents the evaluation of the service identification accuracy of the n-gram method [15] and proposed methods. The n-gram method is significantly time-consuming and could not be applied for our evaluation. Therefore, we modified the test data for the n-gram method to be lightweight and evaluated the performance with this *lightweight dataset*. We explain the lightweight dataset for the n-gram method in Section 4.2. Approximate estimations of time to complete the evaluations with the original dataset by the n-gram method also is discussed in Section 4.2.

4.1 Experimental Setup

We captured the packets of the IP flows in two seasons. The Google and Yahoo services' traffic were captured in September and December 2018. MSN's ones were in Oct. and Dec. 2020. We evaluated the identification accuracies using these captured traffics. The setup of the n-gram method is as follows. N of n-gram is two and n-gram is 2-gram. The threshold of clustering into the same cluster is 0.95. Every group contains ten TLS sessions at most. This setup is the same as in Ref. [15].

For capturing the traffic data, we accessed each service 100 times in each season. The accesses were executed with Mozilla Firefox 52.2. In every TLS session, the version of TLS was 1.2. The SNI field was not encrypted. The captured data of 100 accesses are split into those of 90 accesses and 10 accesses. The former and latter were used for training and testing, respectively. We evaluated the accuracies without cross-validation. This is because the n-gram method is too time-consuming to be evaluated with cross-validation. For reference, we evaluated only the proposed method with Google data with cross-validation. The results are described in Appendix A. The probability of each service was uniform in Sections 4.1, 4.2, and 4.3. That is, $P(A)$s are the same values. The probability obeyed the Zipf's law [27] in Section 4.6. Every service was accessed by executing the web browser in a command line terminal with an option that specifies the URL. All the packets were captured without filtering.

The methods were evaluated as follows. If a method identifies the service correctly, its accuracy is counted as 1; otherwise, it is counted as 0. If a method narrows down the candidate services to n services and the correct service is included in them, it is counted as $1/n$. If a method narrows down the candidates to n and the correct one is not included in them, it is counted as 0. For example, in the case, the candidate services are A, B, C, D, and E, and the correct one is A, the scores are determined as follows. If a method identified the service as A, it is counted as 1 because it is a correct identification. If it identifies as B, it is counted as 0 because it is an incorrect identification. If it narrows down the candidates to two services and identifies as A or B, it is counted as 1/2. The identification result A or B includes the correct one, thus this identification is partially correct. This result is worse than result A, which is a completely correct one, and better than result B, which is an incorrect one. The score of the result A or B (1/2) is between that of A (1) and that of B (0). If a method identifies as A, B, or C, this result is also partially correct. The result is better than an incorrect one like B, and worse than the result with two candidates like A or B. Therefore, its score (1/3) is between 0 and 1/2. If a method does not identify the service, the result is A, B, C, D, or E, and its score is 1/5. This is a partially correct identification and better than an incorrect identification like B. However, the result is not highly valued and its score is around the score of an incorrect one.

The target sites are chosen from sites that are frequently accessed and provide many services. The rank of the frequently accessed sites was published in Alexa Top 100. The chosen sites meet these two conditions. On these sites, the services cannot be identified based on IP addresses and port numbers. The site used a set of IP addresses for services and the address dynamically

changed. We then cannot find a relationship between IP addresses and services. In these services, the port number 443 and the number did not help with identification. All the communications were encrypted in TLS. Except for the limited non-encrypted fields in Handshake could not be analyzed.

4.2 Lightweight Dataset for the n-gram Method

This subsection explains a lightweight dataset for the n-gram method. Before this explanation, we show an estimated time to complete our evaluation by the n-gram method with the original dataset.

As shown in Section 2.3, the n-gram method counts the frequencies of every n-gram in the non-encrypted part of the TLS handshake and creates the n-gram frequency database. It then calculates the correlation coefficient between two vectors' n-gram frequencies for clustering a TLS session. The number of calculated correlation coefficients is proportional to the number of TLS sessions that are already registered in the n-gram frequency database. The time to create an n-gram frequency database is proportional to the square of the number of TLS sessions to be registered, i.e., $o(n^2)$. In the case of 2-gram, the number of kinds of 2-grams is 65,536 and a vector of frequencies has 65,536 elements. Calculation of two vectors of 2-gram frequencies takes a non-short time.

We reduced the size of the Google 15 services' traffic and measured the time to cluster them. In the case of reduced traffic, every service's traffic contains 20 accesses. All this traffic contains 6,948 TLS sessions. It took two days, three hours, and 47 minutes to cluster all these TLS sessions on our computer. We can estimate it takes around 45 days to cluster the non-reduced Google 15 service' traffic in which every service contains 90 TLS sessions with an assumption of $o(n^2)$. This was too hard for us and had to reduce the size of datasets for the n-gram method. We reduced the size of datasets for n-gram in our experiments in Sections 4.2, 4.3, and 4.4. The reduced datasets of Google 15 services and Yahoo 10 services contained 10 accesses and 10 accesses in each service for the training data and testing data, respectively. The reduced dataset of MSN six services contained five accesses and five accesses in each service for training and testing. The original datasets contained 90 accesses and 10 accesses in each service for the training and testing data, respectively.

4.3 Google Services

We first present the evaluation using the 15 Google services in **Table 1** in this subsection. **Figure 7** shows the average identification accuracies where the horizontal axis represents the season of capturing packets, and "Average" represents the average of the accuracies in two seasons. The method based on n-gram is the existing method (n-gram method) [15]. The methods based on hostname in SNI in the first session and based on SNI occurrence are the proposed methods. If the proposed method is given an occurrence vector that is not registered in the SNI occurrence database, it does not identify the service and the accuracy is counted as 1/15.

The results in the figure show that the accuracies of the proposed methods (the orange and gray bars) are higher than that

Table 1 Candidate Google Services.

Name of Service	Hostname in URL
Google Account	myaccount.google.com
Google Calendar	calendar.google.com
Google Document	docs.google.com
Google Drive	drive.google.com
Google Mail	mail.google.com
Google Map	www.google.com
Google Photo	photos.google.com
Google Play	play.google.com
Google Plus	plus.google.com
Google Scholar	scholar.google.com
Google Sheets	docs.google.com
Google Translate	translate.google.com
Google Web Search	ww.google.com
YouTube	www.youtube.com

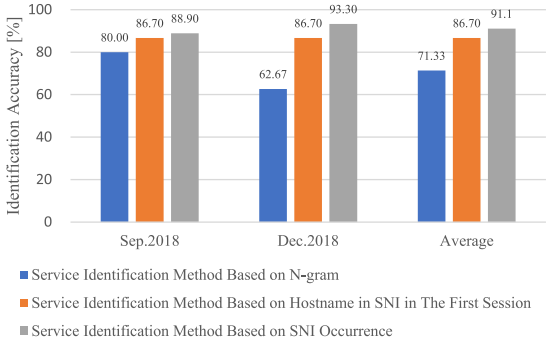


Fig. 7 The Identification Accuracy with 15 Google Services.

Table 2 Candidate Yahoo Services.

Name of Service	Hostname in URL
Yahoo Account	login.yahoo.com
Yahoo Advertising	avertising.yahoo.com
Yahoo Celebrity	www.yahoo.com
Yahoo Finance	finance.yahoo.com
Yahoo Mail	mail.yahoo.com
Yahoo News	www.yahoo.com
Yahoo Web Search	search.yahoo.com
Yahoo Smart TV	smarttv.yahoo.com
Yahoo Sports	sports.yahoo.com
Yahoo Lifestyle	www.yahoo.com

of the existing method (the blue bars). Especially, that of the method based on SNI occurrence was the highest in all the cases. The average ratio of inaccurate identification was decreased by 54%, i.e., from 28.67% to 13.30%, by the method based on the hostname in the SNI in the first TLS session. It was decreased by 69%, i.e., from 28.67% to 8.9%, by the method based on SNI occurrence.

4.4 Yahoo Services

We also evaluated the methods with the ten Yahoo services in **Table 2**. **Figure 8** depicts the identification accuracies. The accuracies of the proposed methods are the highest on average. Although the accuracy of the proposed methods was slightly less than that of the n-gram method in the case of the captured packets in September 2018, that of the proposed method was significantly higher in the case of the packets in December 2018. The average ratio of inaccurate identification was decreased by 72%.

4.5 MSN Services

We evaluated the methods also with the six MSN services in

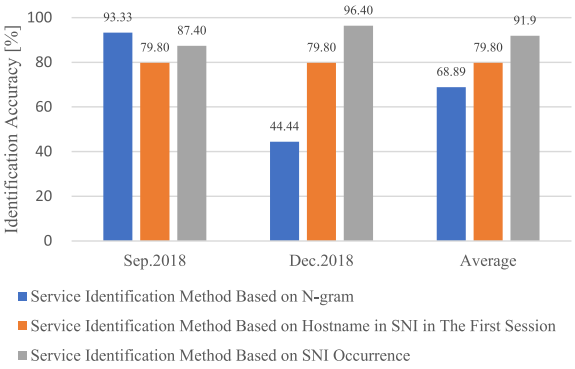


Fig. 8 The Identification Accuracy with Ten Yahoo Services.

Table 3 Candidate MSN Services.

Name of Service	Hostname in URL
MSN Account	login.live.com
MSN Mail	outlook.live.com
MSN Map	www.bing.com
MSN News	www.msn.com
MSN Search	www.bing.com
MSN Store	www.microsoft.com

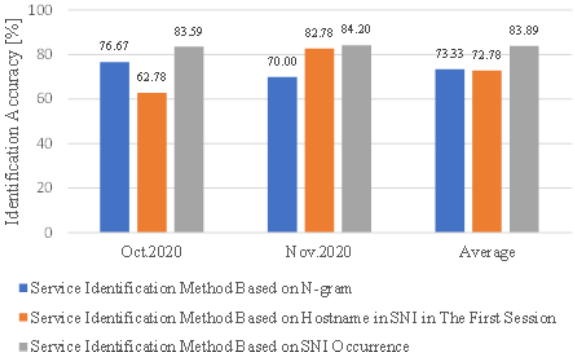


Fig. 9 The Identification Accuracy with Six MSN Services.

Table 3. **Figure 9** shows the identification accuracies. The accuracy of the SNI occurrence method was the highest on average. In particular, the difference between them with the traffic on Oct. 2020 is significant. The average ratio of inaccurate identification was decreased by 41%.

4.6 Google Services with Zipf's Law

This subsection evaluates the methods in an assumption that the probabilities of occurrences of services obey Zipf's law. **Table 4** describes the number of accesses in the testing data. This ranking of services was defined subjectively by ourselves. The detailed explanations of Zipf's law and the number of accesses in the table are presented in Appendix A.2.

Figure 10 shows the results of the SNI occurrence method. The other methods do not depend on the probabilities of occurrences of services and the same result will be obtained with and without this assumption. These results indicate that the results of the SNI occurrence method with and without the assumption were quite similar. These imply that the SNI method works well regardless of the balance of occurrence probability. Exactly writing, the method works slightly well with a biased probability.

4.7 Overhead

Here, we evaluate the consumed time for identification by every method. **Figure 11** shows the average consumed time to identify the service. We identified the 15 Google services using the data captured in Aug. 2019. We identified the packets of 10 accesses for each service. The experimental setup is described in **Table 5**. We measured the time of identification in a virtualized environment. Naturally, the time to identify will be remarkably shorter if a user uses the proposed method in a host operating system.

The average identification time of the existing method was remarkably larger than those of the proposed methods. We can say that the existing method is much harder to be applied in many cases. That of the method based on hostname in SNI in the first session was less than that of the method based on SNI occurrence. In this aspect, the method based on hostname achieved better performance even though its accuracy was less than that of the method based on SNI occurrence. Thus, in case the identification time is important, the method based on hostname in SNI in the first session is a suitable way.

Table 4 The number of accesses in the testing data and the ranking of services.

Google 15 services	number of accesses	
	Uniform distribution	Zipf's law (Zipfian distribution)
account	100	452
calendar	100	226
document	100	151
drive	100	113
gmail	100	90
map	100	75
news	100	65
photo	100	57
play	100	50
plus	100	45
scholar	100	41
search	100	38
sheets	100	35
translate	100	32
youtube_home	100	30

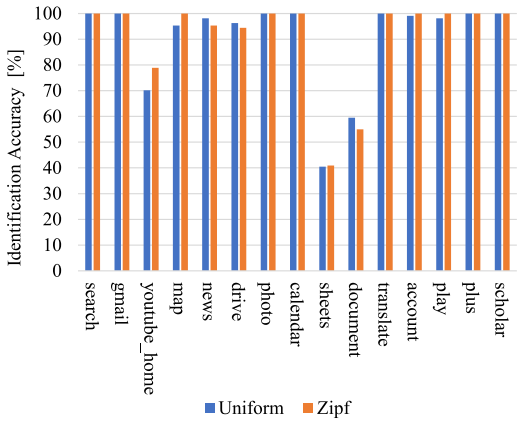


Fig. 10 The Identification Accuracy with 15 Google Services with a probability obeying the Zipf's low.

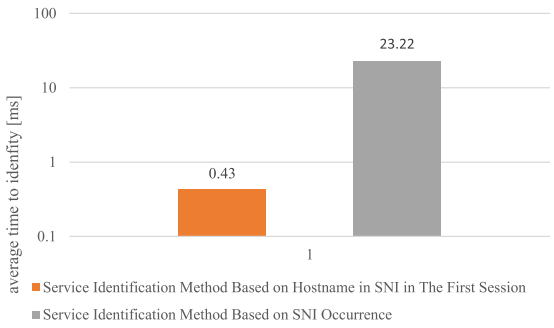


Fig. 11 The Identification Accuracy with Six MSN Services.

Table 5 Specification of the experimental computer.

CPU (host)	**AMD Ryzen 7 3700X 8-Core 3.6 GHz**
Memory Size (host)	**32 GB**
OS (host)	**Windows 10 (64bit) 20H2**
Virtualization system	**Oracle VM Virtualbox 6.1.14**
CPU (guest)	**AMD Ryzen 7 3700X 8-Core 3.6 GHz**
Memory Size (guest)	**4 GB**
OS (guest)	**CentOS 7.9.2009 (Core) , Linux 3.10.0-1160.el7**

For reference, we implemented the n-gram method in Python and compared the time. The comparison is described in Section 5.

5. Discussion

First, we discuss the reason why the proposed methods, especially the method based on SNI occurrence, achieved an accuracy higher than that of the existing method based on the n-gram. This existing method does not analyze field values with protocol analysis but analysis all the bytes in the header including the random values and creates an n-gram database. Therefore, this existing method cannot focus on information that is important for identifying its service and the random values must decrease the accuracy. On the contrary, the proposed method based on SNI occurrence extracts important information [25] by analyzing protocol data units and trains using this information. As a result, this method can identify the service with higher accuracy.

These behaviors affected the stability of the accuracies of the methods. As shown in Figs. 7, 8, 9 and 10, the accuracies of the proposed methods did not strongly depend on the capturing season. However, that of the n-gram method did depend on the season in Fig. 8. One of the reasons is that the n-gram method analyzed also random values. The effect of these random values made the accuracy unstable.

Second, we compare the two proposed methods. The reason why the method based on SNI occurrence outperformed the method based on hostname is that the method based on hostname cannot distinguish services if they have the same hostname in the accessing URL. On the other hand, the method based on SNI occurrence compares the SNIs in all the TLS sessions. In most cases, these SNI occurrence vectors are not the completely same and the method based on SNI occurrence can distinguish services in many cases.

Third, we discuss the possibility of decreasing effectiveness over time. We expect that the effectiveness of the proposed methods will decrease as the implementation of service is updated involving a change of used services and SNIs. This decrease can be relieved by updating the training data. As shown in Section 4, the proposed method, especially the method based on SNI occurrence, can achieve enough accuracy regardless of the seasons of capturing. This implies that the methods can keep enough accuracy by updating the training data.

Fourth, we discuss an avoiding and disrupting method for the proposed methods. The proposed methods identify the services using SNIs. Thus, change in occurring SNIs can decrease their accuracies. However, we expect the methods can suitably identify service with small revision if the mainly used services are the same and some dummy servers are connected. In the work of Ref. [28], it was demonstrated that a method with ignoring non-important SNI increased the accuracies. Thus, the negative effect of connecting useless servers for adding dummy SNIs can be relieved by ignoring non-important SNIs. This ignoring is expected to be useful for relieving the decrease in effectiveness over time if the mainly connected servers are the same.

Fifth, we discuss the service that the proposed method identified with high and low probabilities. The proposed method could not identify the service from flows of Google Documents, Google Sheet, Youtube_home, and Yahoo Finance while this method could identify highly accurately for the other services. In the cases of Google Documents and Google Sheet, their occurring SNIs are the same. Consequently, these services cannot be identified suitably by this method. The numbers of TLS sessions of each SNI of these services also are very similar. Thus, these cannot be identified highly accurately even if the method is improved for taking the number of each SNI into account. We think that this is the limitation of this approach. In the case of Google Youtube_home and Yahoo finance, the numbers of kinds of occurring SNIs are much larger and keep increasing. We discussed this problem in Refs. [29], [30]. We investigated the occurring SNIs and showed that many of these SNIs are automatically generated [29]. We then proposed to cluster these automatically generated SNIs into one group [30]. The evaluation demonstrated that this newly proposed method with clustering improved the accuracies of identifications for Google YouTube_home.

Sixth, we discuss the target accuracy. We think the accuracies in Figs. 7, 8, and 9 of the SNI occurrence method are sufficient in some cases, such as providing higher priority to some important services in a heavily congested condition. In such conditions, non-100% accuracy also useful by improving the quality of services. In addition, inaccurate identification does not cause a severe problem.

Seventh, we discuss the time to identify a service by the n-gram method. For reference, we implemented the n-gram method. It was not implemented in the Click Modular Router but a Python program. The Python program is only for the evaluation of the identification time and does not have a function for switching. The average time of identification of the n-gram method was 456 s, which was around 2,000 times larger than that of the SNI occurrence method. These times cannot be fairly compared because one is implemented in C++ language and the other is implemented in Python. However, the 2,000 times difference cannot be justified by the difference between Python and C++. Thus, we can expect that the identification time of the proposed methods are significantly less than that of the n-gram method.

Eighth, we discuss the data size for evaluation. In our evaluation, we accessed every service 100 times. The larger the data size is, the smaller the bias and noise are. In the work of Ref. [29], the authors surveyed the relationship between the number of accesses and their distribution. The result in it implied that the distributions with 100 data and more data were not quite different. The distribution saturated hundreds of accesses. We then expect that similar results will be obtained if we evaluate the methods with larger data.

Ninth, we discuss the limitation of the proposed methods. As we described, the proposed methods are based on SNI and cannot identify services if the occurring SNIs are completely the same. In our evaluation, Google Document and Google Sheets cannot be distinguished due to this issue. In the case of giving high priority to important services in a disaster, this issue is expected not to be important because both Document and Sheets are less important than some rescue or communication applications. In the case of zero-rating services, we think this is not a serious limitation because most of the existing zero-rating services are not provided with this granularity.

Finally, we discuss a way of implementation and the covered area of the proposed methods. The proposed methods can be implemented in network elements. The programmability of networking devices [31] is increasing, and network switches can be easily controlled using these methods. Especially, a programmable application switch [32] based on DPN provides the way for the implementation of complex functions in network devices based on payload analysis (e.g., DPI).

Our methods cannot be used for communications with TLS 1.3. However, most current communications are using TLS 1.2 or less [33]. Thus, our methods are effective now and for some years in the future. For TLS 1.3, another study on investigation of the ability to cluster sessions of each field in non-encrypted part in TLS 1.3 like [25] is required.

6. Conclusion

In this paper, we focus on service identification of IP flows from the captured packets data that are encrypted with TLS 1.2 or less. We then proposed two methods based on hostname in SNI and SNI occurrence. Our evaluation showed that our methods could identify the service more accurately than the existing method which was based on n-gram of the non-encrypted part of TLS sessions.

In the future, we plan to evaluate our methods with a wider variety of services providers.

Acknowledgments This work was supported by JSPS KAKENHI Grant Numbers 21K11854 and 21K11874. This work was partly supported by National Institute of Information and Communications Technology (NICT), JAPAN.

References

[1] Server Name and Transport Protocol Port Number Registry, available from ⟨http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml⟩ (accessed 2022-10-30).

[2] Moore, A.W. and Papagianaki, K.: Toward the Accurate Identification of Network Applications, *Passive and Active Network Measurement 2005*, Lecture Notes in Computer Science, Vol.3431, pp.41–54, Springer, Berlin, Heidelberg (2005).

[3] Yamauchi, H., Nakao, A., Oguchi, M., Yamamoto, S. and Yamaguchi, S.: Service Identification Based on SNI Analysis, *IEEE Consumer Communications & Networking Conference* (*CCNC 2020*), Work-in-progress paper (2020).

[4] Yamauchi, H., Nakao, A., Oguchi, M., Yamamoto, S. and Yamaguchi, S.: A Study on Service Identification Based on Server Name Indication Analysis, *2019 7th International Symposium on Computing and Networking Workshops* (*CANDARW*), short paper, pp.470–474, DOI: 10.1109/CANDARW.2019.00089 (2019).

[5] Gigis, P., Calder, M., Manassakis, L., Nomikos, G., Kotronis, V., Dimitropoulos, X., Katz-Bassett, E. and Smaragdakis, G.: Seven years in the life of Hypergiants' off-nets, *Proc. 2021 ACM SIGCOMM 2021 Conference* (*SIGCOMM '21*), pp.516–533, Association for Computing Machinery, DOI: 10.1145/3452296.3472928 (2021)

[6] Iwai, T. and Nakao, A.: Identification of Mobile Applications via In-Network Machine Learning Using N-gram for Application-Specific Traffic Control, *IEICE Technical Report*, Vol.115, No.209, NS2015-78, pp.41–46 (Sep. 2015) (in Japanese).

[7] Iwai, T. and Nakao, A.: Adaptive mobile application identification through in-network machine learning, *2016 18th Asia-Pacific Network Operations and Management Symposium* (*APNOMS*), pp.1–6, DOI: 10.1109/APNOMS.2016.7737226 (2016).

[8] Velan, P., Čermák, M., Čeleda, P. and Drašar, M.: A survey of methods for encrypted traffic classification and analysis, *Int. J. Netw. Manag.*, Vol.25, No.5, pp.355–374 (Sep. 2015).

[9] Qualys, Inc.: HTTP Client Fingerprinting Using SSL Handshake Analysis, available from ⟨https://www.ssllabs.com/projects/client-fingerprinting/⟩ (accessed 2020-10-30).

[10] P0f, available from ⟨http://lcamtuf.coredump.cx/p0f3/⟩ (accessed 2020-10-30).

[11] Holz, R., Braun, L., Kammenhuber, N. and Carle, G.: The SSL landscape: A thorough analysis of the x.509 PKI using active and passive measurements, *Proc. 2011 ACM SIGCOMM IMC '11*, pp.427–444 (2011).

[12] Husák, M., Čermák, M., Jirsík, T. and Čeleda, P.: HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting, *EURASIP J. Inf. Secur.*, Vol.2016, No.1, article No.30 (Dec. 2016).

[13] Iwai, T. and Nakao, A.: Identification of Mobile Applications via In-Network Machine Learning for Application Specific QoS Traffic Control, *IEICE Technical Report*, Vol.114, No.477, NS2014-260, pp.487–492 (Mar. 2015) (in Japanese).

[14] Hara, M., Nirasawa, S., Nakao, A., Oguchi, M., Yamamoto, S. and Yamaguchi, S.: Fast Application Identification Based on DPI N-gram, *2016 IEEE 17th Int. Conf. High Performance Switching and Routing Workshop Prog.* (June 2016).

[15] Hara, M., Nirasawa, S., Nakao, A., Oguchi, M., Yamamoto, S. and Yamaguchi, S.: Service Identification by Packet Inspection based on N-grams in Multiple Connections, *2016 4th Int. Symp. Comp. Netw.* (*CANDAR*), pp.686–690, DOI: 10.1109/CANDAR.2016.0123 (2016).

[16] Shbair, W.M., Cholez, T., Francois, J. and Chrisment, I.: A Survey of HTTPS Traffic and Services Identification Approaches, arXiv preprint arXiv:2008.08339, DOI: 10.48550/arXiv.2008.08339 (2020).

[17] Bortolameotti, R., Peter, A., Everts, M.H. and Bolzoni, D.: Indicators of malicious SSL connections, *Network and System Security*, pp.162–175, Springer (2015).

[18] Shbair, W.M., Cholez, T., Goichot, A. and Chrisment, I.: Efficiently bypassing SNI-based HTTPS filtering, Integrated Network Management (IM), *2015 IFIP/IEEE International Symposium*, pp.990–995, IEEE (2015).

[19] Kim, S.-M., Goo, Y.-H., Kim, M.-S., Choi, S.-G. and Choi, M.-J.: A method for service identification of SSL/TLS encrypted traffic with the relation of session ID and Server IP, *2015 17th Asia-Pacific Network Operations and Management Symposium* (*APNOMS*), pp.487–490, DOI: 10.1109/APNOMS.2015.7275373 (2015).

[20] Ding, R. and Li, W.: A hybrid method for service identification of SSL/TLS encrypted traffic, *2016 2nd IEEE International Conference on Computer and Communications* (*ICCC*), pp.250–253, DOI: 10.1109/CompComm.2016.7924703 (2016).

[21] Yang. Y., Kang, C., Gou, G., Li, Z. and Xiong, G.: TLS/SSL Encrypted Traffic Classification with Autoencoder and Convolutional Neural Network, *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems* (*HPCC/SmartCity/DSS*), pp.362–369, DOI: 10.1109/HPCC/SmartCity/DSS.2018.00079 (2018).

[22] Shbair, W.M., Cholez, T., Francois, J. and Chrisment, I.: A multi-level framework to identify HTTPS services, *NOMS 2016: 2016 IEEE/IFIP Network Operations and Management Symposium*, pp.240–248, DOI: 10.1109/NOMS.2016.7502818 (2016).

[23] Barut, O., Zhu, R., Luo, Y. and Zhang, T.: TLS Encrypted Application

Classification Using Machine Learning with Flow Feature Engineering, *2020 10th International Conference on Communication and Network Security* (*ICCNS 2020*), pp.32–41, Association for Computing Machinery, DOI: 10.1145/3442520.3442529 (2020).
[24] Finsterbusch, M., Richter, C., Rocha, E., Muller, J. and Hanssgen, K.: A Survey of Payload-Based Traffic Classification Approaches, *IEEE Communications Surveys & Tutorials*, Vol.16, No.2, pp.1135–1156, DOI: 10.1109/SURV.2013.100613.00161 (2014).
[25] Yamauchi, H., Nakao, A., Oguchi, M., Yamamoto, S. and Yamaguchi, S.: Clustering TLS Sessions Based on Protocol Fields Analysis, *COMPSAC 2018*: *42nd IEEE Comp. Soc. Sign. Conf. Computers, Software & Applications* (2018).
[26] Yamauchi, H., Nakao, A., Oguchi, M., Yamamoto, S. and Yamaguchi, S.: A Study on Clustering Sessions of TLS based on Upload Message, *15th Int. Conf. IP + Opt. Netw.* (*iPOP2019*), P-9 (2019).
[27] Gabaix, X.: Zipf's Law for Cities: An Explanation, *The Quarterly Journal of Economics*, Vol.114, No.3, pp.739–767, DOI: 10.1162/003355399556133 (1999).
[28] Asaoka, R., Nakao, A., Oguchi, M. and Yamaguchi, S.: Choosing SNIs based on their Occurrence Probability on Service Identification, *IEICE Technical Report*, Vol.122, No.105, NS2022-45, pp.94–99 (2022) (in Japanese).
[29] Soma, Y., Nakao, A., Oguchi, M., Yamamoto, S., Yamaguchi, S. and Kobayashi, A.: Ocurring SNIs for Servicve Indetification, *2020 IEEE 9th Global Conference on Consumer Electronics* (*GCCE*), pp.586–587, DOI: 10.1109/GCCE50665.2020.9292038 (2020).
[30] Soma, Y., Nakao, A., Yamamoto, S., Oguchi, M., Yamaguchi, S. and Kobayashi, A.: Service Identification of TLS Connections based on SNI Analysis, *IEICE 2020 International Conference on Emerging Technologies for Communications* (Dec. 2020).
[31] Kohler, E., Morris, R., Chen, B., Jannoti, J. and Kaashoek, M.F.: The Click Modular Router, *ACM Trans. Comp. Syst.* (Aug. 2000).
[32] Kanaya, T., Yamauchi, H., Nirasawa, S., Nakao, A., Oguchi, M., Yamamoto, S. and Yamaguchi, S.: Intelligent Application Switch Supporting TCP, *IEEE Int. Conf. Cloud Netw.* (Oct. 2018).
[33] Qualys, Inc.: Qualys SSL Labs - SSL Pulse, available from ⟨https://www.ssllabs.com/ssl-pulse/⟩ (accessed 2022-09-29).
[34] Adamic, L.A. and Huberman, B.A.: Zipf's law and the internet, *Glottometrics*, Vol.3, No.1, pp.143–150 (2002).

Appendix

A.1 Comparison of Results with and without Cross-validation

In the case of the Google 15 services, the accuracies of the proposed method with and without cross-validation were 91.95% and 92.80%, respectively.

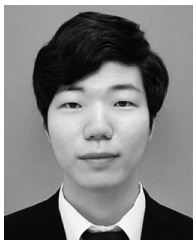
A.2 Zipf's Law

Zipf's law is an empirical law of a relation between the frequency of occurrence of an event and its rank when the events are ranked concerning the frequency of occurrence [34]. The most frequent one is ranked as the first one. The Zipf's law equation states that:

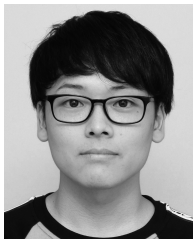
$$f(r) = \frac{C}{r^{\alpha}}$$

where r is the rank, $f(r)$ is the frequency of occurrence of the event at rank r, and C is a constant. α is 1 in usual cases. Zipf's Law was originally concerned with word frequency. It stated that the frequency of a word is inversely proportional to its rank.

In this paper, we assume that the relation between the frequency of access to a service and its rank obeys this law. In the case of Table 4, C is 452. The frequency of the r-th service was approximately C/r. The order of this ranking was defined by the authors.



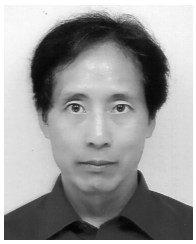
Ryo Asaoka received his B.E. degrees in Engineering Kogakuin University in 2022. He now stays in Kogakuin University to study electrical engineering and electronics.



Yuto Soma received his M.E. degrees in Kogakuin University in 2021.

Hiroaki Yamauchi received his M.E. degrees in Kogakuin University in 2020.

Akihiro Nakao received his Ph.D. degree in Computer Science from Princeton University. He is currently a Professor of the Applied Computer Science Course at the Interfaculty Initiative in Information Studies, Graduate School of Interdisciplinary Information Studies, The University of Tokyo.



Masato Oguchi received his Ph.D. degrees from The University of Tokyo in 1995. In 1995, he was a Researcher at the National Center for Science Information System (NACSIS), currently known as National Institute of Informatics (NII). From 1996 to 2000, he was a Research Fellow at the Institute of Industrial Science, University of Tokyo. From 2000 to 2003, he was an Associate Professor at the Research and Development Initiative, Chuo University. He joined Ochanomizu University as an Associate Professor in 2003 and becomes Professor since 2006. His areas of interest include network computing middleware, high-performance computing, and mobile networking.

Saneyasu Yamaguchi received Engineering Doctor's degree (Ph.D.) at Tokyo University in 2002. During 2002–2006, he stayed in Institute of Industrial Science, the University of Tokyo to study I/O processing. He now with Kogakuin University. Currently his researches focus on operating systems, virtualized systems, and storage system.

Aki Kobayashi received his Ph.D. degree in Engineering from Tokyo Institute of Technology in 2000. He is now with Kogakuin University. His current research interests include information recommendation, information retrieval, image recognition, and interactive system.