Certificate-Aware Encrypted Traffic Classification Using Second-Order Markov Chain

Meng Shen*, Mingwei Wei*, Liehuang Zhu*, Mingzhong Wang†, and Fuliang Li‡
*Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications School of Computer Science, Beijing Institute of Technology, Beijing, P. R. China
†Faculty of Arts and Business, University of the Sunshine Coast, Queensland, Australia
‡College of Information Science and Engineering, Northeastern University, P. R. China
{shenmeng, weimingwei, liehuangz}@bit.edu.cn; mwang@usc.edu.au; lifuliang@ise.neu.edu.cn

***Abstract*—With the prosperity of network applications, traffic classification serves as a crucial role in network management and malicious attack detection. The widely used encryption transmission protocols, such as the Secure Socket Layer/Transport Layer Security (SSL/TLS) protocols, leads to the failure of traditional payload-based classification methods. Existing methods for encrypted traffic classification suffer from low accuracy.**

In this paper, we propose a certificate-aware encrypted traffic classification method based on the Second-Order Markov Chain. We start by exploring reasons why existing methods not perform well, and make a novel observation that certificate packet length in SSL/TLS sessions contributes to application discrimination. To increase the diversity of application fingerprints, we develop a new model by incorporating the certificate packet length clustering into the Second-Order homogeneous Markov chains. Extensive evaluation results show that the proposed method lead to a 30% improvement on average compared with the state-of-the-art method, in terms of classification accuracy.

I. Introduction

Network traffic always acts as the carrier of web service, network communications, as well as network attacks. Traffic classification is widely used to classify traffic belonging to specific network applications from mass flows. Systems with the traffic classification function are often deployed in gateway devices, which audit flows passing through the gateway to effectively intercept malicious traffic and ensures normal network operations.

Traditional traffic classification technologies are often designed based on the analysis of the packet contents [17, 21], such as the port-based methods and the payload-based methods. These methods can only fulfil efficient recognition under unencrypted traffic. With the wide use of encryption protocols, such as the SSL/TLS protocol, the traditional traffic classification methods lose efficacy. Therefore, it is desirable to develop classification methods for encrypted traffic [6]. Maciej et al. propose such a method based on first-order homogeneous Markov chains [11], which is the state-of-the-art method in encrypted traffic classification. They take advantage of a sequence of message types in the SSL/TLS header of a given application, which appears in a single direction flow from a server to a client, to build the first-order homogeneous Markov chain as a statistical fingerprint for a given application.

The precondition, for the above encrypted traffic classification method to separate applications from one another, lies in that the fingerprint for each application should be distinctive enough. Unfortunately, it is quite difficult to get distinctive fingerprints based on the first-order Markov chain, since states for modeling are limited. When similar state transitions appear in multiple application fingerprints, a traffic flow is always categorized as the application with the maximum likelihood. In other words, traffic flows corresponding to a Markov chain with low probability are prone to be misclassified. For instance, we create first-order Markov chain fingerprint for Instagram based on datasets that we collect, more than 75% of the SSL/TLS sessions generate the same state transition chain. While the rest Instagram chain instances with low probabilities are prone to be misclassified as other applications, which results in a high False Negative rate. Similar results are also observed for Amazon EC2 flows [11].

To address these problems, we propose a new method to classify encrypted traffic. The basic idea is to increase the diversity of application fingerprints. On the one hand, we use the second-order homogeneous Markov chain, instead of the first-order one, for comprehensively considering time and space complexity, veracity of model and difficulty of implementation. We take advantage of 3 states to build state transition when modeling the second-order Markov chain, which captures more distinctive features of a given application. On the other hand, we focus on the message type of `Certificate` in SSL/TLS sessions. Our analysis reveals that certificate packet length can be regarded as a critical feature for traffic classification. Thus, we build more distinctive fingerprints for given applications, by incorporating certificate packet length into the second-order Markov chain. The newly proposed method results in a significant improvement over the state-of-the-art method, in terms of the accuracy of application discrimination. The major contributions of this paper are summarized as follows.

We analyze large amounts of application traffic and point out the observations that lead to low accuracy of the state-of-the-art method. When similar state transition appears in different application fingerprints, that is to say, there are similar SSL/TLS sessions for different applications, traffic classification becomes harder, which always results in incorrect traffic classification and high false negative rate. We find two reasons for above situation, one is that state transition

composed of only 2 states can hardly distinguish different chains, another is that features for modeling containing only message type are so limited. The fact is also the weakness of state-of-the-art method, we demonstrate this defect does exist by experiments.

We propose a new method to build application fingerprints by incorporating certificate clustering into the second-order homogeneous Markov chains. In the second-order Markov chain, two states ahead are considered when forecasting the current state, which is more accurate than the first-order one. Moreover, we employ certificate packet length of SSL/TLS session as a key feature of applications. The probability of different applications having same certificate packet length is not high, we can regard certificate packet length as an auxiliary means of application identification. We cluster all application certificates in datasets by K-Means and design an evaluation metric of clustering results to determine how many clusters there should be. In order to create more states for modeling, we incorporate the certificate states into the second-order Markov chain, which serves as the fingerprint for traffic classification.

We verify the effectiveness of the proposed method by real traffic datasets collected from 12 representative applications in two campus sites. These datasets contain 24-hour real traffic traces. We select 12 application traces for evaluation, which consist of about hundreds of thousands of flows and millions of packets in total. The category of applications is abundant, including finance (Alipay, Yirendai), social contact (Facebook, Github, Instagram, LinkeIn, Twitter, Weibo), O2O (Online to Offline) [20] service (Ele), music (NeteaseMusic) and life assistant (Baidu, Evernote). Evaluation of our method is based on these datasets. The evaluation results show a good accuracy of about 90% for traffic classification, and a 30% improvement on average compared with the state-of-the-art method.

To the best of our knowledge, this is the first paper that considers the certificate packet length of SSL/TLS sessions in encrypted traffic classification field. The rest of paper is organized as follows. We summarize related work and describe our motivation in Section II and Section III, respectively. The modeling progress of our method for encrypted traffic classification is presented in Section IV, which is followed by the design details of certificate clustering in Section V. Section VI exhibits the evaluation results of our method. And the paper is concluded in Section VII.

II. Related Work

A. SSL/TLS Background

The Secure Sockets Layer [9] and its successor Transport Layer Security [7, 16] protocols provide communication security and data integrity over the Internet. Therefore, SSL/TLS is primarily used to encrypt confidential data sent over an insecure network. In the network protocol structure, SSL/TLS is located between application layer and transport layer, encrypting and transferring upper-layer data to lower-layer. SSL/TLS includes two sub-protocols. One is Handshake Protocol with the function of negotiating parameters of an SSL/TLS session. The other is Record Protocol with the function of transferring encrypted data under secure parameters generated from Handshake Protocol. In our study, we extract effective features mainly from Handshake Protocol.

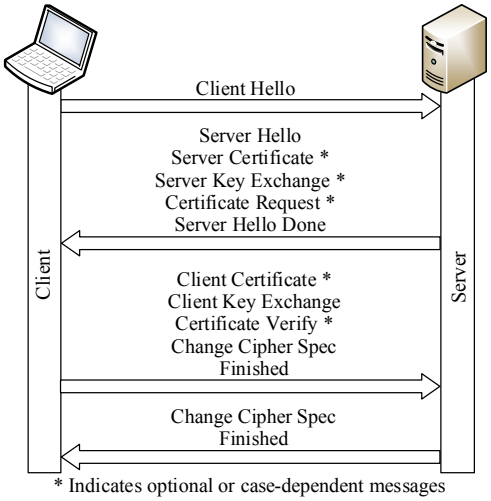

Fig. 1. Interaction of SSL/TLS Handshake Protocol

Figure 1 presents an example interaction of SSL/TLS Handshake Protocol. Initially, client sends request for encryption communication to server labeled as `Client Hello`, which includes four attributes: Protocol Version, Random Number-1, Cipher Suite and Compression Method. After receiving that, server responses `Server Hello`, which includes four attributes: Protocol Version confirmed, Random Number-2, Cipher Suite confirmed and Server Certificate. Next, client responses Random Number-3, `Change Cipher Spec` and client `Finished` message if server certificate passes validation. The two sides share three random numbers so far and use them as parameters to generate same session key with method negotiated in advance. Ultimately, server responses `Change Cipher Spec` and server `Finished` message. After this, the Handshake Protocol stage is over, the following communication is protected by encryption and compression method negotiated before.

TABLE I
Notations and their corresponding SSL/TLS message types

Notation	SSL/TLS Message Type
20	Change Cipher Spec
21	Alert
22:2	Server Hello
22:11	Certificate
22:12	Server Key Exchange
22:13	Certificate Request
22:14	Server Hello Done
22:17	Encrypted Handshake Message
22:18	New Session Ticket
22:20	Finished
23	Application Data

In consideration of client individual configurations, such as the SSL/TLS protocol settings in different web browsers,

we expect a little differences in client-side models, whereas server-side model should be coincident all the time. So in this paper, we consider only server-side message types of an SSL/TLS session when modeling. We use compact notation [11] to represent message types during SSL/TLS session (cf. Table I). For instance, we denote `Server Hello` by 22:2 and `Application Data` by 23:.

B. Summary of Traffic Classification Methods

Methods under unencrypted network traffic environment. The basic idea of these method is to excavate useful information from packet contents such as port number and payload. The port-based method implements traffic classification by checking TCP/UDP port number on transport layer protocol [14]. In the early stage of Internet, applications use specific port assigned by IANA (The Internet Assigned Numbers Authority) to set up communication. For instance, thunder is always combined to port number 8000 and tomcat is combined to 8080. Unfortunately, with the popularity of port jump and random port technology, the port-based methods lose effect. The typical approach of payload-based methods is to compare traffic payload content with feature set built in advance, then mark matching results as the basis of classification. For instance, feature strings of HTTP protocols include `GET` and `POST`, meanwhile `Tencent-VQ` is the feature string of QQ Voice. The precision and efficiency of payload-based method is influenced by feature matching algorithm, which includes one-mode and multi-mode. One-mode refers to one scan can only match one feature string, such as Knuth-Morris-Pratt algorithm [10] and Boyer-Moore algorithm [4]. Multi-mode refers to one scan can match a group of feature strings, such as Aho-Corasick algorithm [1], Commentz-Walter algorithm [5]. However, with the popularity of encryption transmission protocols, the payload-based methods also lose effect.

Methods under encrypted network traffic environment. Bernaille et al. propose a method to detect applications in SSL encrypted connections only using the size of the first few packets of an SSL connection [3]. Maciej et al. propose a method to create application fingerprints by modeling sequences of message types observed in server-side during SSL/TLS sessions based on first-order homogeneous Markov chain [11]. This method is state-of-the-art so that we recommend in following text. Experiments prove that this method can be used to classify encrypted traffic.

A sequence of states $X_1, ..., X_T$ represents a chain of server-side message types in a SSL/TLS session. X_t is a discrete-time random variable, which $t = t_0, t_1, ..., t_n \in T$. The value of X_t is marked as i_t, which is either an SSL/TLS message type (e.g. 22:11) or a series of SSL/TLS message types transmitted in one TCP segment (e.g. 22:2,20:). In homogeneous first-order Markov chain, a state transition from time $t-1$ to t is invariant.

$$P\left(X_t{=}i_t|X_{t-1}{=}i_{t-1}\right) = P\left(X_t{=}j|X_{t-1}{=}i\right) = p_{i-j}. \quad (1)$$

In addition, they define q_{i_t} as ENter Probability Distribution (ENPD), which represents probability of i_t for the first state of Markov chain. They also define w_{i_t} as EXit Probability Distribution (EXPD), which represents probability of i_t for the last state of Markov chain. Based on these defines, the probability of a sequence of states $X_1, ..., X_T$ representing a SSL/TLS session flow is as follows.

$$P\left(\{X_1, ..., X_T\}\right) = q_{i_1} \times \prod_{T}^{t=2} p_{i_{t-1}-i_t} \times w_{i_T}. \quad (2)$$

A larger probability above means that a given sequence of message types of SSL/TLS session is closer to fingerprint model of application traffic.

III. Motivation

To build and valid fingerprints based on first-order Markov chain, we collect two datasets gathered on specific routers. The traffic collection scheme is described as Figure 2, clients interact with given application servers through a specific router, which connects campus network with the Internet. We capture traffic flows through router and execute additional operations (e.g. wipe out non-SSL/TLS encrypted traffic) to generate datasets. The *Campus1* and *Campus2* datasets come from two specific routers located in different sites in a same campus and both of these two datasets contain 24 hours long trace starting on November 10, 2015.

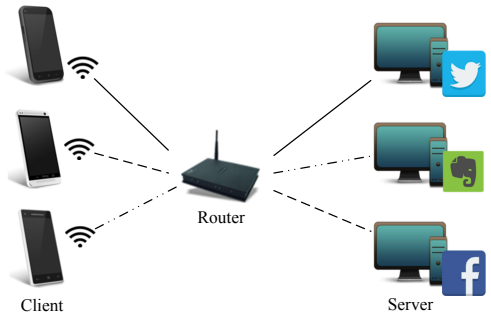

Fig. 2. Framework of Traffic Collection

Based on above datasets, we built first-order Markov chain fingerprints for given applications as in [11]. Although these fingerprints contribute to classifying encrypted traffic, it fails in some cases. For instance, Figure 3 and 4 illustrate the fingerprints for LinkedIn and Evernote, respectively. For clarity, the diagrams contain only states with meaningful probabilities to be simplified. In fact, full Markov chain fingerprint models are generally complex for including lots of states. A sequence of SSL/TLS message types `22:2 - 22:11 - 22:12 - 20:,22:17 - 23` is an Evernote flow which we have known its ground truth. Plugging this sequence into above two fingerprints, we find the probability of being a LinkedIn flow is equal to 0.5611, while being an Evernote flow is equal to 0.1146. We incorrectly mark this sequence as a LinkedIn flow for higher probability, which results in a low true positive rate (TPR) of Evernote and a high false positive rate (FPR) of LinkedIn.

The above problem is not individual, it also appears in fingerprints of Instagram, Twitter and other applications. After

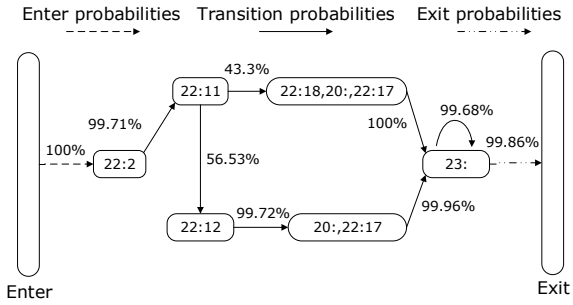

Fig. 3. Parameters of the Fingerprint for LinkedIn

in-depth study, we find two reasons below. Firstly, we consider transition in only two discrete states, which can not describe the features of the flow well. When state transitions of which a specific sequence of SSL/TLS message types consists are all appeared in many fingerprints of different applications, we consider this sequence is a flow corresponding max fingerprint matching probability. The error of this method to classify traffic is high. Secondly, states used for modeling are only derived from message types of SSL/TLS session in current approach. After statistics, we find the message types are very limited, which results in small quantity of states used for modeling and further leads to low discrimination of fingerprints.

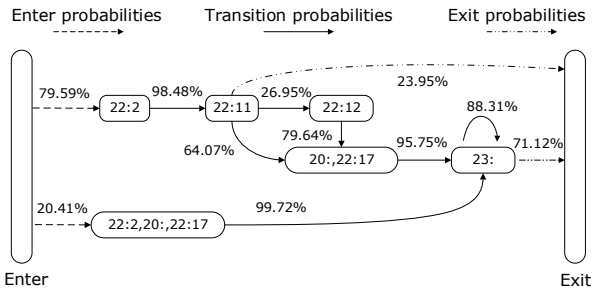

Fig. 4. Parameters of the Fingerprint for Evernote

Solution Overview. We intend to address the above problems in two aspects as follows.

1. We use second-order Markov chain instead of first-order one after comprehensive consideration of implementation difficulty, time cost and space cost. In this way, we can ease the problem that first-order Markov chain describe characteristics of fingerprint badly to a certain degree.
2. We analyze server certificate and find it is available to enrich states for modeling. Server certificate is used to indicate server identity and authenticate visitor identity, it consists of issuers, subjects, certification paths, valid dates and so on. Therefore, the length of server certificates are usually different due to different certificate content. For modeling, we combine the message type of `Certificate` and its length to generate composite states to enrich the state sets and increase discrimination of fingerprint.

IV. Modeling Progress

In this section, we mathematically formulate the process of modeling based on second-order Markov chain and the procedure of incorporating the certificate states into fingerprints.

A. Second-order Homogeneous Markov Chain

We propose a method based on second-order homogeneous Markov chain to model message types in SSL/TLS session.

Supposing discrete-time random variable X_t for any $t = t_0, t_1, ..., t_n \in T$, the corresponding value $i_t \in \{1, ..., s\}$, where i_t represents a message type (e.g. 22:11) in an SSL/TLS session or a sequence of message types transmitted in one TCP segment (e.g. 22:2,20:).

For formalization, we assume that X_t represents a second-order Markov chain [13, 15, 18], i.e. considering preceding two states to estimate the current state.

$$\begin{aligned} &P\left(X_t = i_t | X_{t-1} = i_{t-1}, X_{t-2} = i_{t-2}, ..., X_1 = i_1\right) \\ =&P\left(X_t = i_t | X_{t-1} = i_{t-1}, X_{t-2} = i_{t-2}\right) \end{aligned}. \quad (3)$$

Ulteriorly, we assume the second-order Markov chain is homogeneous, i.e. a state transition from time $t-2$ to time $t-1$ and again to time t is invariant.

$$\begin{aligned} &P\left(X_t = i_t | X_{t-1} = i_{t-1}, X_{t-2} = i_{t-2}\right) \\ =&P\left(X_t = k | X_{t-1} = j, X_{t-2} = i\right) = p_{i-j-k} \end{aligned}, \quad (4)$$

with the state transition matrix of second-order homogeneous Markov chain:

$$P = \begin{bmatrix} p_{1-1-1} & p_{1-1-2} & \cdots & p_{1-1-s} \\ & & \cdots & \\ p_{1-s-1} & p_{1-s-2} & \cdots & p_{1-s-s} \\ & & \cdots & \\ p_{s-1-1} & p_{s-1-2} & \cdots & p_{s-1-s} \\ & & \cdots & \\ p_{s-s-1} & p_{s-s-2} & \cdots & p_{s-s-s} \end{bmatrix}, \quad (5)$$

where $\sum_{k=1}^{s} p_{i-j-k} = 1$. Furthermore, we denote the ENter Probability Distribution (ENPD) [11] as the probability to enter second-order Markov chain with the first two states, which is defined in Equation (6),

$$Q = \left[q_{1-1}, ..., q_{1-s}, q_{2-1}, ..., q_{2-s}, ..., q_{s-1}, ..., q_{s-s}\right], \quad (6)$$

where $q_{i-j} = P(X_{t+1} = j, X_t = i)$ at time t_0, and we define the EXit Probability Distribution (EXPD) [11] as the probability to exit Markov chain with the last two states, which is defined in Equation (7),

$$W = \left[w_{1-1}, ..., w_{1-s}, w_{2-1}, ..., w_{2-s}, ..., w_{s-1}, ..., w_{s-s}\right], \quad (7)$$

where $w_{i-j} = P(X_t = j, X_{t-1} = i)$ at time t_n. Based on the above definitions, the probability that a sequence of states $X_1, ..., X_T$ representing a SSL/TLS session flow is as follows:

$$P\left(\{X_1, ..., X_T\}\right) = q_{i_1 - i_2} \times \prod_{t=3}^{T} p_{i_{t-2} - i_{t-1} - i_t} \times w_{i_{T-1} - i_T}. \quad (8)$$

We are hard to demonstrate fingerprint by state transition plane graph similar to Figure 3 and 4, since application fingerprints based on second-order Markov chain are three-dimensional. We design one format file to manifest specific application fingerprint as Figure 5 presented. The first line

represents total state number of Markov chain that we assume to be s and the second line displays all state names. The following $s \times s$ lines represent state transition matrix and the implications of matrix's rows and columns are corresponding to state names presented in second line. Moreover, the following s lines represent ENPD matrix and the last s lines represent EXPD matrix.

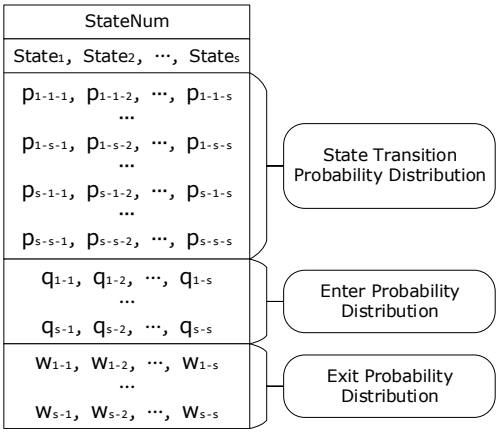

Fig. 5. Fingerprint Format of Second-order Markov Chain

For better describing the process of application fingerprint generation, we illustrate it with following example of several sequences of session message types observed in server-side SSL/TLS flow of Evernote application.

`22:2 - 22:11 - 20:,22:17 - 23:`

`22:2 - 22:11 - 20:,22:17 - 23: - 23: - 23: - 21:`

`22:2,20:,22:17 - 23: - 23:`

The ENPD, EXPD and probabilities of state transition are based on frequencies observed in the sequences. After statistics for the above example, we get results as follows. The ENPD vector contains two elements $P_{22:-22:11} = 0.67$ and $P_{22:2,20:,22:17-23:} = 0.33$. The EXPD vector contains three elements $P_{20:,22:17-23} = 0.33$, $P_{23:-21:} = 0.33$ and $P_{23:-23:} = 0.33$. In probability vector of state transition, $P_{22:2-22:11-20:22:17} = 1$, $P_{23:-23:-23:} = 0.5$, $P_{23:-23:-21:} = 0.5$ and $P_{22:11-20:,22:17-23:} = 1$. Application fingerprint model consists of these probabilities. Based on model, we can calculate the probability that SSL/TLS session representing the given application flow (cf. Equation (8)). For instance, the probability of following SSL/TLS session message type sequence `22:2,20:,22:17 - 23: - 23: - 23:` being an Evernote flow is equal to: $P(\{X_1, ..., X_4\}) = 0.05445$.

B. Consideration of Certificate

Besides message types of SSL/TLS session, we additionally consider certificate packet length when creating states for modeling. We extend original single state of `Certificate` to several states of `Sub-Certificate`, where extension criterion depends on its length. In this way, we enrich the state set and further make fingerprint more distinctive.

Supposing there are c clusters after clustering certificate packet length of all applications in training dataset, we define $C = \{cent_1, ..., cent_c\}$ as cluster center set, where $cent_i$ represents value of cluster center length. We define:

$$cent = \underset{cent_i}{\operatorname{argmin}} |length - cent_i|, i \in \{1, ..., c\}, \quad (9)$$

as cluster center which a given certificate packet length is subjected. Based on above defines, the probability that certificate packet length distribution of a specific application is as follows:

$$F_{app} = [f_{app,1}, ..., f_{app,c}], \quad (10)$$

where $f_{app,i}$ represents the probability that certificate packet length of a given application *app* subjects to cluster *i*. In our study, several different applications are discussed, we generate certificate packet length distribution matrix:

$$F = \begin{bmatrix} f_{1,1} & \cdots & f_{1,c} \\ \vdots & \cdots & \vdots \\ f_{r,1} & \cdots & f_{r,c} \end{bmatrix}, \quad (11)$$

where $\sum_{j=1}^{c} f_{i,j} = 1, i \in \{1, ..., r\}$, r represents the number of application class discussed, which equals to 12 in our case. Combining to second-order Markov chain we discussed above, we can find the probability that a sequence of SSL/TLS session message types being a specific application flow is:

$$M(\{X_1,..., X_T\}) = P(\{X_1,..., X_T\}) \times F_{app\_idx, clut\_idx}, \quad (12)$$

where *app_idx* represents a label of specific application and *clut_idx* represents a label of cluster that application is subjected. A larger value of $M(\{X_1, ..., X_T\})$ means the sequence of SSL/TLS session message types is closer to fingerprint of given application.

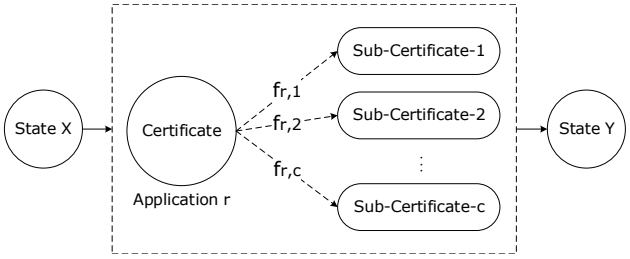

Fig. 6. Process of Incorporating Certificate Clustering into Markov Chain

For better describing the process of incorporating certificate clustering into Markov chain, we illustrate it in Figure 6. State X and State Y represents states of Markov chain, we replace single state of `Certificate` with the dashed box we extend. For computing state transition probability in second-order Markov chain, we ignore content of dashed box first and still regard it as single `Certificate`. Afterwards, we need to multiply above result by the probability that which `Sub-Certificate` the specific `Certificate` is subjected to. The `Sub-Certificate` corresponds to the certificate packet length clustering result.

V. Certificate Packet Length Clustering

In the previous section, we introduce how we combine certificate packet length with second-order Markov chain to improve application fingerprint grandly. In this section, we discuss certificate packet length clustering algorithm in detail.

Certificate packet length clustering is the basis to extend original single state of `Certificate`. For certificate packet length set of all applications, there is no training pattern and standard for clustering, therefore, unsupervised clustering meets our requirement. In our case, content of data set for clustering is certificate packet length, which is one-dimensional. We use K-Means algorithm [19] for its simplicity and utility.

K-Means is a popular clustering algorithm and the basic idea is to find a partition solution by iteration, with minimum total error by using cluster means to replace samples. The time complexity of K-Means is $\mathcal{O}(tlk)$ (t indicates iterations, l indicates total elements, k indicates cluster numbers) and space complexity is $\mathcal{O}(l+k)$. It begins by selecting k objects as cluster centers labeled $cent_i$ arbitrarily and initializing cluster center set labeled C with those k $cent_i$. Then repeat following two steps until no change in C or iteration times are equal or greater than given threshold labeled *th*. Step 1 is assigning all data objects to their closest cluster center to complete partition once. Step 2 is reassigning new cluster centers as mean value of the data objects in each cluster. Thus we complete K-Means algorithm preliminarily.

Although the idea of K-Means is simple, there are two factors have large influence on cluster effect as follows.

- K-Means algorithm is sensitive to the selection of the initial cluster centers. Selecting different cluster centers often results in different clustering effects.
- The number of clusters k is decided by user. However, users can hardly know distribution of data objects well in general.

In our case, we solve above problems with two custom-made solutions below. These two solutions are significant parts of our certificate packet length clustering algorithm, which presents in Algorithm 1. The time complexity of Algorithm 1 is $\mathcal{O}(ijtlk)$ (i indicates iterations, j indicates frequency of calling K-Means, $\mathcal{O}(tlk)$ is time complexity for K-Means) and space complexity is $\mathcal{O}(i+j+l+k)$.

For selection of cluster centers, we initialize them for many times and select the best one. For each initialization, we select cluster centers arbitrarily. Note that each value of cluster center should among the minimum and maximum of data objects. In order to assess clustering effect based on current initialization, we denote the Sum of Squared Error (SSE) as criterion, which is defined in Equation (13).

$$SSE = \sum_{j=1}^{n} \|length_j - lc_j\|^2, \tag{13}$$

where lc_j represents the cluster center that $length_j$ is subjected. In other words, SSE represents the sum of the squares of distance between each certificate packet length sample and its corresponding cluster center [12]. For the given data object set L, cluster number k and iteration threshold th, we use them as parameters to process K-Means algorithm circularly and calculate SSE for each circulation. We break the loop until the minimum of SSE is in a stable range. A smaller value of SSE means a better effect of clustering. In this way, we select appropriate initial cluster centers and ensure the good clustering effect.

Algorithm 1 Certificate Packet Length Clustering

Require: $L = \{length_1, ..., length_n\}, th$
Ensure: $C = \{cent_1, ..., cent_k\}$
1: Initialize $k \leftarrow 2$
2: **for** $k \geq 2$ **do**
3: **repeat**
4: $C, LC \leftarrow KMeans(L, k, th)$
5: $SSE \leftarrow \sum_{j=1}^{n} \|length_j - lc_j\|^2$
6: **until** minimum of SSE is in a stable range
7: Calculate $Wt \leftarrow [w_1, ..., w_r]$
8: Calculate $F_i \leftarrow [f_{i,1}, ..., f_{i,k}]$
9: $SPV \leftarrow \sum_{i=1}^{r} w_i \times \sigma^2(f_{i,j}) \times (k-1), j \in \{1, ..., k\}$
10: **if** SPV reaches a peak in a stale range **then**
11: return
12: **end if**
13: $k \leftarrow k + 1$
14: **end for**

For selection of cluster number, we enumerate k from 2 to larger number and design judge rule to find appropriate one. For a given k, we are not difficult to get appropriate cluster centers by processing K-Means algorithm above. Further, we can calculate the probability F_{app} that certificate packet length of a specific application subjected to each clusters (cf. Equation (10)). We hope that the distribution of elements of F_{app} vector is the more discrete the better. If distribution is uniform, the cluster of certificate packet length loses its significance. Variance is used to reflect discrete degree of random variable. Thus, we denote Sum of Probability Variance variant (SPV) as criterion measuring effect of cluster number selection, which is defined in Equation (14).

$$SPV = \sum_{i=1}^{r} w_i \times \sigma^2(f_{i,j}) \times (k-1), j \in \{1, ..., k\}, \tag{14}$$

where w_i represents the weight of application i certificate packet length in all certificates, $f_{i,j}$ refers to Equation (11) and $\sigma^2(f_{i,j})$ represents the agonic variance of $f_{i,j}$. Analysis shows that when the number of clusters we given is close to the real number of clusters, the value of SPV access peak value in a stable range, but when the number of clusters we given is less than the real, the value of SPV is rising in a sharp angle. In this way, we are able to determine the appropriate value of k.

VI. Evaluation

In this section, we are dedicated to evaluating the effectiveness of the proposed method.

A. Preliminary

Methods in Comparison. We refer to the method proposed in this paper as Second-Order Markov chain fingerprint considering Certificate packet length (SOCRT). In order to present a comprehensive understanding on the effects of the Second-Order Markov chain and the certificate packet length clustering, we leverage three other methods for comparison, which are summarized as follows:

- First-Order Markov chain fingerprint (FOM), which is analogous to the state-of-the-art method for encrypted traffic classification [11].
- Second-Order Markov chain fingerprint (SOM), which replaces the First-Order Markov chain in FOM with the Second-Order Markov chain.
- First-Order Markov chain fingerprint considering certificate packet length (FOCRT), which enriches the modeling states of FOM by considering the certificate packet length clustering described in Section V.

Datasets. The preparation work of datasets is presented in Section III, including framework design, source selection, duration of traffic collection and so on. We generate *Campus1* and *Campus2* datasets, where *Campus1* includes 45785 flows (521337 packets) and *Campus2* includes 48201 flows (512911 packets). In our case, we mainly pay attention to the information in SSL/TLS handshake phase, so there are great majority of SSL/TLS handshake sessions in our datasets.

Ground Truth and Application Selection. Ground truth is a standard to verify methods for traffic classification. In this paper, we establish the ground truth by two steps. The first step is to find domain name by parsing IP address of a flow. In this process, we utilize an open web service named *Whois* [8], which is a tool for searching the corresponding domain names of given IP. The second step is to extract and analyze some particular strings so that we can confirm which application the flow belongs to. Table II shows several particular strings in domain names identified specific applications.

TABLE II
PARTICULAR STRINGS IN DOMAIN NAMES TO IDENTIFY APPLICATIONS

Application	Strings in Domain Names
Alipay	*.alipay.com
Baidu	*.baidu, *.bdstatic
Ele	*.ele.me
Evernote	*.evernote, *.yinxiang
Facebook	*.facebook.*, *.fb*
Github	*.github
Instagram	*.instagram, *.igcdn, *.igsonar
LinkedIn	linkedin.*
NeteaseMusic	music.163.*
Twitter	*.twitter.com
Weibo	*.weibo.com
Yirendai	*.yirendai

The approach to establish ground truth is not universally valid. One constraint is that we can not obtain ground truth if *Whois* fails to resolve IP address into domain address. Another constraint is that we are unable to collect all signatures of application domain names, which result in that some application flow miss its ground truth.

Our traffic classification method is universal and valid under any circumstance, since handshaking is indispensable in SSL/TLS. To verify effectiveness for our method, we need to know ground truth in advance. So we set some limitations of application selection in our experiment evaluation process, we only choose applications that easy to establish ground truth by *Whois* tools and domain name signature resolution.

Table III presents the number of flows and packets of each application in training dataset. For each training dataset, we capture at least thousands of flows, about ten thousands of packets, to ensure fingerprint that we modeling can cover all features of corresponding application.

TABLE III
FLOWS AND PACKETS OF APPLICATIONS

	Campus1		Campus2	
Applications	Flows	Packets	Flows	Packets
Alipay	5173	42706	6030	35661
Baidu	2208	24904	2617	23929
Ele	2906	15835	2898	15846
Evernote	5069	27581	5537	24873
Facebook	3604	192135	3794	189229
Github	2315	21894	2247	21953
Instagram	4066	21796	4250	21237
LinkedIn	5547	55626	4964	54580
NeteaseMusic	3519	27084	3494	27440
Twitter	4055	41089	3808	40778
Weibo	4359	23277	5496	29982
Yirendai	2964	27410	3066	27403

Cross-validation. For the purpose of validating application fingerprints that created by different methods in this paper, we take advantage of two heterogeneous datasets (*Campus1* and *Campus2*) to establish a 2-fold cross-validation. More specifically, we create Markov chain and clustering certificate packet length to establish fingerprint based on training dataset and validate fingerprints on remaining dataset. When we use *Campus1* as training dataset, the *Campus2* is used to validate, inversely, when *Campus2* is used as training dataset, *Campus1* has to be testing dataset.

The process of validation is as follows. We extract application flows from testing dataset in advance, that is we get ground truth beforehand. The ground truth is as a standard to measure the result of classifier. Decision process of classifier in this paper is based on Maximum Likelihood criterion [2], it is a multi-hypothesis decision problem. Suppose twelve hypothesis $H_i, i = 1, ..., 12$ represents each application we modeling in our experiment, we consider classifier result is hypothesis that the sequence Y is most likely:

$$H = \underset{H_i}{\operatorname{argmax}} L(\{Y_1, ...Y_T\}|H_i) \tag{15}$$

where $L(\{Y_1, ..., Y_T\})$ represents the likelihood of message sequence $\{Y_1, ...Y_T\}$ under each hypothesis. $L(\{Y_1, ..., Y_T\}) \equiv P(\{X_1, ..., X_T\})$, which represents probability of message sequence calculated under each application fingerprint based

on first or second-order Markov chain only; $L(\{Y_1, ..., Y_T\}) \equiv M(\{X_1, ..., X_T\})$, which represents probability of message sequence calculated under each fingerprint based on incorporating certificate packet length clustering methods.

Criteria of Cross-validation. In our experiment evaluation, it is necessary to provide a reliable criteria for validation of application fingerprint. We consider three meaningful metrics to evaluate the effect of different methods: TPR, FPR and a hybrid formula combined TPR and FPR. True Positive (TP) means the validation result is consistent with ground truth, i.e. one sequence of message types in SSL/TLS session is correctly identified. For instance, a Twitter flow is correctly classified as Twitter. False Positive (FP) means one sequence of message types in SSL/TLS session is incorrectly classified as other application. For instance, a LinkedIn flow is classified as Evernote. To more visibly reveal effect of methods, we denote Fractional combination of TPR and FPR (FTF) as a tidy criteria, which is defined in Equation (16).

$$FTF = \sum_{i=0}^{r} w_i \frac{TPR_i}{1 + FPR_i}, \tag{16}$$

where r represents number of applications in this paper, i.e. $r = 12$, w_i means the weight of application i. Higher TPR and lower FPR declare a better classification decision. Thus, the value of FTF is proportional to classification method effect.

B. Effect of the Second Order Markov Chains

Table IV and Table V present the 2-fold cross-validation results of comparison of FOM and SOM. In comparison with FOM, SOM results in better effect, where weighted TPR rises about 13% and weighted FPR falls 12%.

TABLE IV
COMPARISON OF FOM AND SOM.TRAINING DATASET:CAMPUS1, VALIDATION DATASET:CAMPUS2

	FOM		SOM	
Applications	TPR	FPR	TPR	FPR
Alipay	0.2730	0.5596	0.8127	0.3838
Baidu	0.8609	0.1746	0.9013	0.1784
Ele	0.8138	0.6703	0.8097	0.6115
Evernote	0.3602	0.4075	0.3738	0.2322
Facebook	0.9329	0.0567	0.9057	0.0233
Github	0.9589	0.0129	0.9931	0.0449
Instagram	0.8047	0.3402	0.8468	0.0659
LinkedIn	0.4020	0.3594	0.4022	0.0638
NeteaseMusic	0.9946	0.0034	0.9946	0.0034
Twitter	0.5963	0.6031	0.8175	0.1671
Weibo	0.6630	0.3581	0.9195	0.3178
Yirendai	0.5135	0.0725	0.7426	0.1976
Weighted Mean	0.6316	0.3306	0.7580	0.1923

Taken example of Alipay, its FOM fingerprint consists of multiple state transition chain without a main chain, thus the probability of these chain instances are very low, which is easy to falsely classified as other applications. This fact results in low TPR of Alipay and high FPR of other applications. Under the circumstances of second-order Markov chain, the constraint above is eased. For instance, a Alipay sequence `22:2-22:11-22:18,20:,22:17-23:` is always classified as LinkedIn in FOM, our study finds that the similar flow in LinkedIn is `22:2-22:11-22:18,20:,22:17-23:`$^+$, where `23:`$^+$ means there are more than one `23:` in state transition, like `23:-23:` and `23:-23:-23:`. Obviously, SOM can easy distinguish these two types of flow, while FOM can not.

TABLE V
COMPARISON OF FOM AND SOM.TRAINING DATASET:CAMPUS2, VALIDATION DATASET:CAMPUS1

	FOM		SOM	
Applications	TPR	FPR	TPR	FPR
Alipay	0.2871	0.4293	0.8320	0.3834
Baidu	0.8089	0.1026	0.8846	0.2317
Ele	0.8496	0.5912	0.3823	0.1431
Evernote	0.3715	0.4515	0.3310	0.2697
Facebook	0.9085	0.0749	0.8856	0.0135
Github	0.9065	0.0	0.9444	0.0066
Instagram	0.7753	0.3487	0.8520	0.1114
LinkedIn	0.5606	0.3438	0.9799	0.2616
NeteaseMusic	0.9943	0.0136	0.9937	0.0134
Twitter	0.5993	0.6784	0.8306	0.1915
Weibo	0.7029	0.3063	0.8330	0.2997
Yirendai	0.5153	0.0851	0.7414	0.2226
Weighted Mean	0.6458	0.3136	0.7827	0.2027

In Table V, we find that TPR of Ele falls sharply but TPR of LinkedIn rises drastically from FOM to SOM, while these two rates are stable in Table IV. Analysis shows that they share lots of same sequences of message types, which are hard to distinguished no matter based on FOM or SOM. These same sequence instances classified according to maximum likelihood, even the probabilities calculated under two application fingerprints are so close. Therefore, the TPR of application with a little larger probability calculated based on these sequences is high, while TPR of another application is low. The fact remind us not only pay attention to TPR or FPR of each application, but also focus on global evaluation.

C. Effect of Certificate Packet Length Clustering

Table VI and Table VII present the 2-fold cross-validation results of comparison of FOM and FOCRT. In comparison with FOM, FOCRT results in better effect, where weighted TPR rises about 22% and weighted FPR falls 20%.

There are a certain amount of common flows in Ele and LinkedIn, which can not distinguished by FOM or SOM well, such as `22:2-22:11-22:18,20:,22:17-23:-23:`. But we observe a fact that even though the message type sequences of flows in two applications become indistinguishable, another feature can be leveraged to differentiate these two types of flows. The key feature is certificate packet length of specific application. Distribution of Ele certificate packet length is in scope with centered 1474, while LinkedIn is in range with centered 1054. On the basis of SPV presented in section V, experiments of certificate packet length clustering declare 6 is an appropriate number of clusters in our case. Besides, results show that certificates of Ele and LinkedIn are

TABLE VI
COMPARISON OF FOM AND FOCRT.TRAINING DATASET:CAMPUS1, VALIDATION DATASET:CAMPUS2

	FOM		FOCRT	
Applications	TPR	FPR	TPR	FPR
Alipay	0.2730	0.5596	0.8349	0.1182
Baidu	0.8609	0.1746	0.8632	0.1706
Ele	0.8138	0.6703	0.8734	0.0562
Evernote	0.3602	0.4075	0.8278	0.2887
Facebook	0.9329	0.0567	0.9340	0.0590
Github	0.9589	0.0129	0.9589	0.0129
Instagram	0.8047	0.3402	0.9771	0.0641
LinkedIn	0.4020	0.3594	0.9977	0.1890
NeteaseMusic	0.9946	0.0034	0.9966	0.0
Twitter	0.5963	0.6031	0.6454	0.0456
Weibo	0.6630	0.3581	0.8316	0.1856
Yirendai	0.5135	0.0725	0.7510	0.0663
Weighted Mean	0.6316	0.3306	0.8742	0.1170

in different clusters. Combining this condition to Markov chain fingerprint, we succeed in distinguishing two flows above.

TABLE VII
COMPARISON OF FOM AND FOCRT.TRAINING DATASET:CAMPUS2, VALIDATION DATASET:CAMPUS1

	FOM		FOCRT	
Applications	TPR	FPR	TPR	FPR
Alipay	0.2871	0.4293	0.8360	0.1102
Baidu	0.8089	0.1026	0.8120	0.2298
Ele	0.8496	0.5912	0.8882	0.0777
Evernote	0.3715	0.4515	0.8015	0.2833
Facebook	0.9085	0.0749	0.8864	0.0577
Github	0.9065	0.0	0.9065	0.0
Instagram	0.7753	0.3487	0.9824	0.1010
LinkedIn	0.5606	0.3438	0.9827	0.2249
NeteaseMusic	0.9943	0.0136	0.9983	0.0108
Twitter	0.5993	0.6784	0.6641	0.0413
Weibo	0.7029	0.3063	0.8291	0.1709
Yirendai	0.5153	0.0851	0.7642	0.0713
Weighted Mean	0.6458	0.3136	0.8620	0.1281

But for twitter, FOCRT results in a not good effect. Twitter's certificate packet lengths distribute in cluster where the certificate packet length of Evernote, Facebook and Baidu are also distribute in. In this case, effect of certificate clustering is weaken, we need to find a new algorithm or incorporate other method to fill this loophole. Fortunately, SOM behavior well in similar occasions like classifying twitter flow. Therefore, we consider a hybrid method combining second-order Markov chain and consideration of certificate packet length as follows.

D. Effect of Combining Second Order Markov Chain with Certificate Packet Length Clustering

Table VIII and Table IX present the 2-fold cross-validation results of comparison of FOM and SOCRT. In comparison with the methods above, SOCRT results in the best effect, where weighted TPR rises about 26% and weighted FPR falls about 24%.

Comparing with SOM, SOCRT fills the holes that unsatisfactory classification effect of Ele, Evernote and LinkedIn,

TABLE VIII
COMPARISON OF FOM AND SOCRT.TRAINING DATASET:CAMPUS1, VALIDATION DATASET:CAMPUS2

	FOM		SOCRT	
Applications	TPR	FPR	TPR	FPR
Alipay	0.2730	0.5596	0.8610	0.0334
Baidu	0.8609	0.1746	0.9013	0.1753
Ele	0.8138	0.6703	0.8892	0.0278
Evernote	0.3602	0.4075	0.9106	0.2636
Facebook	0.9329	0.0567	0.9059	0.0225
Github	0.9589	0.0129	0.9931	0.0445
Instagram	0.8047	0.3402	0.9840	0.0568
LinkedIn	0.4020	0.3594	0.9955	0.0290
NeteaseMusic	0.9946	0.0034	0.9966	0.0
Twitter	0.5963	0.6031	0.8284	0.0269
Weibo	0.6630	0.3581	0.9303	0.1486
Yirendai	0.5135	0.0725	0.7500	0.0528
Weighted Mean	0.6316	0.3306	0.9146	0.0757

where certificate consideration plays a part. Comparing with FOCRT, SOCRT solve the problem that certificate packet length clustering does poorly, such as classification of Baidu and Twitter. In other words, SOCRT integrates features of SOM with FOCRT, which results in the most mature method of encrypted traffic classification than any other methods proposed in this paper. The TPR of SOCRT is about 90% and the FPR is only roughly 8%.

TABLE IX
COMPARISON OF FOM AND SOCRT.TRAINING DATASET:CAMPUS2, VALIDATION DATASET:CAMPUS1

	FOM		SOCRT	
Applications	TPR	FPR	TPR	FPR
Alipay	0.2871	0.4293	0.8756	0.0981
Baidu	0.8089	0.1026	0.8854	0.2302
Ele	0.8496	0.5912	0.8989	0.0451
Evernote	0.3715	0.4515	0.8839	0.2571
Facebook	0.9085	0.0749	0.8861	0.0167
Github	0.9065	0.0	0.9444	0.0056
Instagram	0.7753	0.3487	0.9880	0.0968
LinkedIn	0.5606	0.3438	0.9801	0.0374
NeteaseMusic	0.9943	0.0136	0.9977	0.0105
Twitter	0.5993	0.6784	0.8372	0.0248
Weibo	0.7029	0.3063	0.8399	0.1318
Yirendai	0.5153	0.0851	0.7635	0.0591
Weighte Mean	0.6458	0.3136	0.8978	0.0925

More specifically, we design FTF to broadly evaluate each method, which is defined in Equation (16). Figure 7 presents FTF values of 2-fold cross-validation of different methods. Green bar represents effect under circumstance that *Campus1* as training dataset and *Campus2* as testing dataset, while yellow bar is on the contrary. Seen from the figure, SOCRT gains best effect obviously. We propose SOCRT by incorporating certificate clustering into second-order Markov chain. In this way, we greatly enriched the features of application fingerprint and reduces the similarity of different fingerprints, which results in good effect in most cases.

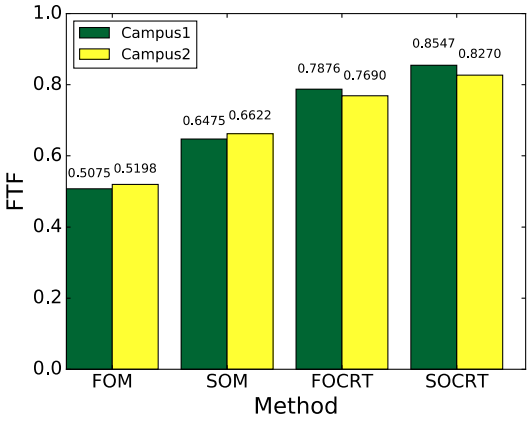

Fig. 7. FTF of Methods

E. Discussion

Although our method leads to good accuracy of traffic classification in most instances, it still results in misclassification in some small probability occasions. For example, `22:2-22:11-22:12,20:,22:17-23` is one kind of primary message type sequence in Yirendai, which happens to be the main kind of flow of Evernote. Since the probability computed with Yirendai fingerprint is lower than that with Evernote, and the certificate packet lengths of Yirendai and Evernote are subject to the same cluster, we falsely classify these Yirendai flows as Evernote under the maximum likelihood criterion. Fortunately, the occurring probability of the above case is so small that our method is still suitable for most situations. Even so, we seek for a further improvement by considering the length of the first Application Data message following the SSL/TLS handshake message. The length varies along with the application in general and we will delve into it in the follow-on work.

Currently, we create fingerprints in offline training stage, while performing traffic classification online. Application fingerprints we created should be evolved periodically to ensure the effect of traffic classification, because old fingerprints can not contain all features with application updates. We defer the design and evaluation of online classification algorithms as the future work.

VII. Conclusion

In this paper, we propose one kind of application fingerprint for encrypted traffic classification. We first identify the reasons why the existing first-order Markov fingerprint is not accurate enough in many cases. To overcome these drawbacks, we propose a new method by incorporating certificate packet length clustering into the second-order Markov chains. To evaluate the effectiveness of our method, we conduct extensive experiments that compares SOCRT with existing methods. Evaluation results reveal that both the second-order Markov chains and the certificate packet length clustering make contributions to improving the accuracy of application discrimination, and SOCRT achieves the best effects.

In future work, we plan to further investigate other features in encrypted traffic to dig inspiration and improve the classification accuracy.

Acknowledgments

This work has been supported in part by National Science Foundation of China (61100172, 61272512), Beijing Natural Science Foundation (4132054, 4164098). Prof. Liehuang Zhu is the corresponding author.

References

[1] A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[2] J. Aldrich et al. Ra fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3):162–176, 1997.

[3] L. Bernaille and R. Teixeira. Early recognition of encrypted applications. In *Passive and Active Network Measurement*, pages 165–175. Springer, 2007.

[4] R. S. Boyer and J. S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.

[5] B. Commentz-Walter. *A string matching algorithm fast on the average*. Springer, 1979.

[6] A. Dainotti, A. Pescape, and K. C. Claffy. Issues and future directions in traffic classification. *Network, IEEE*, 26(1):35–40, 2012.

[7] T. Dierks. The transport layer security (tls) protocol version 1.2. 2008.

[8] P. T. Endo and D. F. H. Sadok. Whois based geolocation: A strategy to geolocate internet hosts. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 408–413. IEEE, 2010.

[9] A. Freier, P. Karlton, and P. Kocher. The secure sockets layer (ssl) protocol version 3.0. 2011.

[10] D. E. Knuth, J. H. Morris, Jr, and V. R. Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350, 1977.

[11] M. Korczynski and A. Duda. Markov chain fingerprinting to classify encrypted traffic. In *INFOCOM, 2014 Proceedings IEEE*, pages 781–789. IEEE, 2014.

[12] L. I. Kuncheva and D. P. Vetrov. Evaluation of stability of k-means cluster ensembles with respect to random initialization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(11):1798–1808, 2006.

[13] I. L. MacDonald and W. Zucchini. *Hidden Markov and other models for discrete-valued time series*, volume 110. CRC Press, 1997.

[14] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen. Portvis: a tool for port-based detection of security events. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 73–81. ACM, 2004.

[15] A. E. Raftery. A model for high-order markov chains. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 528–539, 1985.

[16] E. Rescorla. *SSL and TLS: designing and building secure systems*, volume 1. Addison-Wesley Reading, 2001.

[17] F. G. O. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus. Lightweight, payload-based traffic classification: An experimental evaluation. 2008.

[18] A. Shamshad, M. Bawadi, W. W. Hussin, T. Majid, and S. Sanusi. First and second order markov chain models for synthetic generation of wind speed time series. *Energy*, 30(5):693–708, 2005.

[19] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[20] Y. Ye et al. Online to offline food delivery situation and challenges in china. case company ele. me. 2015.

[21] M. Zhang, W. John, K. Claffy, and N. Brownlee. State of the art in traffic classification: A research review. In *PAM Student Workshop*, pages 3–4, 2009.