Webpage Fingerprinting Using Only Packet Length Information

Meng Shen[‡], Yiting Liu[‡], Siqi Chen[‡], Liehuang Zhu[‡], and Yuchao Zhang[†]
[‡] Applications School of Computer Science, Beijing Institute of Technology, Beijing, P. R. China
[†]School of Software Engineering, Beijing University of Posts and Telecommunications, Beijing, P. R. China
shenmeng@bit.edu.cn; {liuyitingbit, siqichennet}@foxmail.com; liehuangz@bit.edu.cn, yczhang@bupt.edu.cn

***Abstract*—Encrypted web traffic can reveal sensitive information of a user, such as their browsing histories. Existing studies on encrypted traffic analysis attacks usually focus on traffic fingerprinting of different websites rather than that of webpages from a same website. Fine-grained webpage fingerprinting allows exploiting more private information of users, e.g., their interests within a news website or an online shopping website. Since webpages from a same website usually have very similar features (e.g., statistical information) that make them indistinguishable, existing solutions may end up with low accuracy.**

In this paper, we propose a novel webpage fingerprinting method based on a simple and comprehensible idea. We make an observation that the length information of packets in bidirectional interaction between clients and servers can be a distinctive feature in webpage fingerprinting. Then, we abstract the cumulative length of a sequence of packets to represent the fingerprint of a specific webpage. More precisely, only the first 100 packets in the loading process of a webpage is considered, thus enabling early-stage fingerprinting. The experimental results with real-world datasets demonstrate that our method is superior to other state-of-the-art approaches in terms of classification accuracy and time complexity. To the best of our knowledge, this is the first work on fine-grained webpage fingerprinting.

***Index Terms*—webpage fingerprinting, encrypted traffic classification, machine learning**

I. Introduction

Website fingerprinting has attracted extensive attention in the field of traffic analysis. It aims at identifying the contents of websites by observing patterns of data flows. Compared with websites, webpage fingerprinting is able to get more valuable and private information. For instance, by identifying the specific webpages accessed by users on shopping websites, eavesdroppers are capable of analyzing users' shopping behaviors and further speculate on user preferences. By identifying the contents of webpages that users browse on news websites, eavesdroppers are able to obtain users interested contents and even analyze the political inclination of users. This is by no means trivial.

Most traditional traffic analysis techniques are based on deep packet inspection, such as payload-based approaches and port-based approaches. In recent years, more and more network traffic have been encrypted by SSL/TLS protocol [4], thus leading the traditional methods to be unusable. Many studies have focused on website fingerprinting using encrypted traffic analysis. For instance, Andriy et al. [12] presented cumulative packet length to identify websites. Saman et al. [5] applied dynamic time warping (DTW) algorithm to classify websites. Compared with webpages, different website traffic has distinct differences and is easier to distinguish. Although several of their work declare that they have the ability to classify webpages, these webpages are distributed in different websites.

Webpages traffic on the same websites have almost the same parameters of SSL/TLS, thus leading their encrypted traffic fingerprints extremely similar to each other. Furthermore, in practical application, fingerprint identification system requires low computational complexity and quick response time. Those intricate fingerprint detection methods need to be avoided. These all bring great challenges to webpage fingerprinting.

In this paper, we propose a fine-grained webpage fingerprinting based on a simple and comprehensible idea. We make an observation that the length information of packets in bidirectional interaction between clients and servers can be a distinctive feature in webpage fingerprinting. To simplify this problem, we set the uplink packet length to 0 and utilize cumulative packet length information to represent the webpage loading process. For each webpage, we only need to consider the first 100 accumulative packets. The proposed approach can be combined with the traditional machine learning algorithms to construct webpage fingerprint classifiers. Since the simplicity and efficiency of our method, we are capable of implementing webpage fingerprinting in early stage.

We summarize our main contributions as follows:

1) Using real-world encrypted traffic, we analyze the inefficiency of other methods [5][12] in identifying webpage fingerprints.
2) We propose a fine-grained webpage fingerprinting method using only packet length information based on the analysis of encrypted traffic. We also combine it with traditional machine learning algorithms and construct webpage fingerprint classifiers.
3) We verify the effectiveness of the proposed method. The experimental results show that the accuracy of our method is up to 91.6%, which is higher than other state-of-the-art methods. More importantly, with low computational complexity, our method is able to implement early-stage webpage fingerprinting.

To the best of our knowledge, this is the first work targeting at fine-grained webpage fingerprinting. This paper is organized as follows. We first review the related work in Section II. Then, we present the motivation of this study and the construction

of webpage fingerprinting in Section III and Section IV, respectively. In Section V, we evaluate the performance of the proposed method, and compare it with other state-of-the-art approaches. We conclude this paper in Section VI.

II. Related Work

A. SSL/TLS Background

In the early stage, HTTP protocol was used to the web service. This protocol uses plaintext transmission and brings many security risks. In order to solve the risk of plaintext transmission, Netscape designed the SSL secure transmission protocol for the Web in 1994, which is the origin of SSL.

SSL/TLS is a protocol that interfaces with the transport layer (such as TCP/IP) and the application layer (such as HTTP)[4][6]. It solves the problem of transmission security through *handshake* protocol and *transport* protocol. The handshake protocol is the process of establishing a connection, this stage uses asymmetric encryption. After this process is completed, a session key will be generated, then, the client and the server use this session key for encrypted communication.

SSL/TLS protocol is theoretically secure, and it can indeed encrypt data transmission content. However, information such as timestamp and packet length of data transmission can still be obtained, as a result many researchers or attackers take this as a breakthrough to analyze encrypted traffic and get information.

B. Summary of Encrypted Traffic Classification Study

Website Classification Methods. Andriy et al. [12] presented a simple and efficient method to classify websites. They used cumulated sum of packet size to represent traffic trace, and extracted a fixed number of discriminative features from traffic traces with different lengths. Support Vector Machine (SVM) was used as classifier in their work. Their method is superior in terms of computational efficiency, while cannot distinguish specific webpage traffic effectively, especially on the same website. Korczynski et al. [10] used first-order Markov chain to classify encrypted traffic based on the characteristics of SSL/TLS session. Shen et al. [14] conducted further research on their scheme. They use second-order Markov chains and application attribute bigrams to classify websites, achieving higher classification accuracy. Saman and Douglas [5] proposed a novel classification method using only packet timing information on the link. They applied DTW algorithm to classify between traffic traces with time sequences. Different from existing methods, their approach does not need to achieve the start/end of web fetches.

Smartphone Applications Classification Methods. With the popularity of smartphones, many researchers have made a lot of studies with the encrypted traffic generated by mobile devices. Vincent et al. [16] proposed a method called *AppScanner* to identify smartphone apps from encrypted network traffic. They use incoming streams, outcoming streams and complete streams to extract 54 statistic features and ranking them by characteristic contributions. Random forests shows the best performance in their experiments, which the accuracy of APP classification reaches 99%. Moreover, their method is proved to be robust and is able to realize real-time identification. Mauro et al. [3] realized the identification of user actions by analyzing Android encrypted network. They used DTW algorithm to calculate the sequence of data packets and extract features. After that, random forest was used to be their classifier. Finally, the classification of user actions is reached by clustering. Fu et al. [7] proposed a new framework by exploiting a multi-label multi-view strategy to classify App service. Fu et al. [8] analyzed packet lengths and time delay to propose a system called CUMMA aiming to detect service usage of mobile messaging app. Ranjan et al. [13] presented approximate matching of persistent LExicon with search-engines. In order to address the mobile application identification problem. Grolman et al. [9] used transfer learning to identify user action in mobile apps by analyzing encrypted traffic.

Although many studies have focused on the analysis of encrypted network traffic, it is still a serious challenge to classify multiple webpages on the same websites, since the traffic generated by webpage is similar. In this paper, we put forward a simple and comprehensible webpage fingerprinting approach. Our scheme has low computational complexity and enables online detection, thus bringing greater advantages in practical applications.

III. Motivation

In this section, we first introduce our dataset in Section III-A. Next, in Section III-B, we consider the limitation of other fingerprinting methods under encrypted traffic, which motivate us to carry out our further study. Finally, we introduce our analysis process of fine-grained webpage fingerprinting in Section III-C.

A. Data Collection

To implement webpage fingerprinting, we need to collect webpage traffic on the same website in real-world. *Jingdong* (JD for short) is one of the largest online shopping mall in China, and its shopping interface of different commodities is very similar. We select JD [1] as our experimental website, and intend to classify the webpages (shopping interface of different commodities) on it.

In our study, all data was captured in *Ali* cloud ECS server with Wireshark. Ali cloud is one of the most popular cloud providers, whose infrastructures cover all regions of China. To simulate the traffic generated by real users accessing webpages, we employ 6 *Ali* cloud ECS servers with Windows system, 4 of which are located in different cities in North China, called North1, North2, North3, North4 and the other 2 cloud servers are located in different cities in East China called East1, East2. The webpage traffic on JD was captured on different cloud servers. We choose Chrome with version number 67.0.3396.99 as the experimental browser.

We made use of crawler technology to pick 100 commodities randomly in JD, and each commodity corresponds to one webpage. Our dataset contains 6000 flows of these 100

webpages, an average of 60 visits per webpage. The process of data collection took about a month and experienced various network environments.

B. Inefficiency of Existing Methods

Traffic fingerprinting requires quick response time and low computational complexity in practical application . As a result, we expect to exploit single and low-complexity features during the process of traffic analysis.

Saman and Douglas [5] put forward the idea of using only timing information to analyze encrypted traffic. They achieved to classify webpages, though these webpages are from different websites. However, timestamp information of the traffic is easily affected by network fluctuations and the fingerprint identification process is time consuming. As a result, we focus on the packet length information and try to find more efficient approaches.

Andriy et al. [12] proposed a simple and comprehensible website fingerprinting attack based on the cumulative packet length of traffic, named CUMUL. Their method is superior in terms of computational efficiency, but it can only be applied to website classification. In order to explore the effect of CUMUL on webpage classification, we process our dataset based on their method.

The traffic is first divided into flows (detailed steps are shown in Section IV-A); Then, packet length sequence $T = (p_1, ..., p_N)$ is extracted from each flow, where $p_i > 0$ represents the downlink packet length and $p_i < 0$ represents the uplink packet length. At this point, the cumulative packet length is expressed as $C(T) = (c_1, ..., c_N)$, where

$$c_i = \begin{cases} p_i, & \text{if } i = 1; \\ c_{i-1} + p_i, & \text{if } 1 < i \leq N. \end{cases} \tag{1}$$

In Fig. 1, we visualize the sequence C from the recording of two websites (JD and Website1) in our dataset. For each website, we randomly select 5 webpages. From Fig. 1 we can see that these two websites have different traffic traces and can be easily distinguished with CUMUL. However, for webpages on the same website, their traffic traces are very similar and intermingled with each other. Therefore, webpages cannot distinguishable with CUMUL.

This result motivates us to carry out our further study. We intend to find a simple and efficient approach to classify webpages on the same website with packet length information based on the analysis of encrypted traffic.

C. Webpage Traffic Analysis

We follow the method that using cumulative packet lengths to represent the traces of flows. In our dataset, for easy observation, we randomly select 5 webpages to describe their cumulative traces. Fig. 2 exhibits the cumulative traces of these 5 webpages (WP). From Fig. 2 we can see, although these flows that belonging to the same website have very similar traces, but between 30 and 60 packets, they are different from each other.

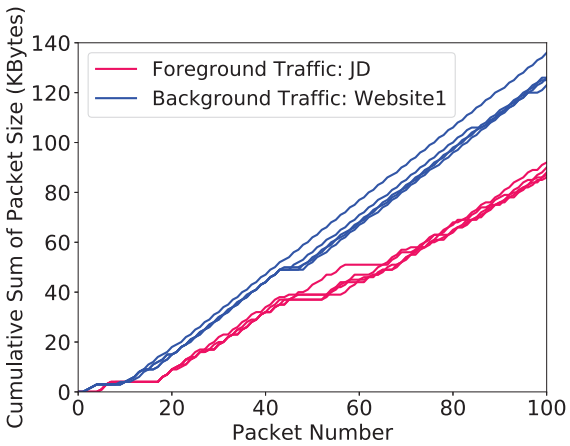

Fig. 1: Fingerprints of two websites derived with CUMUL

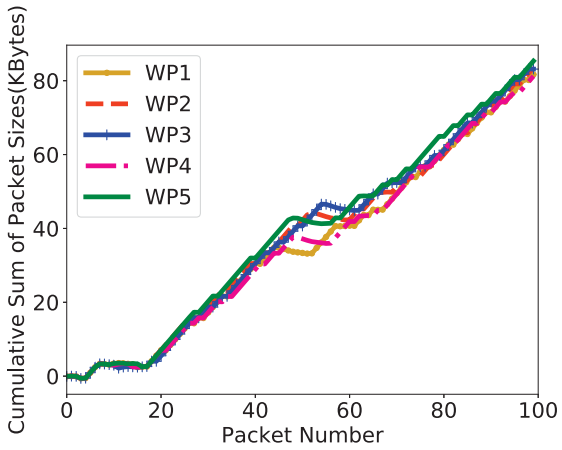

Fig. 2: Cumulative traces of five different webpages

To gain insight into the differences between the packets, we analyze the first 100 uplink and downlink packets interaction information. Ignoring the specific messages of packet during transmission, Fig. 3 shows the typical interactions between the first 100 uplink and downlink packets during the webpages loading process. The interaction process can be divided into 3 stages: (i) alternating transmission of uplink and downlink packets; (ii) only transmitting uplink packets; (iii) alternating transmission of uplink and downlink packets. Our research found that different webpages have different time to enter the second stage, and the second stage may be the key to classifying webpages.

We further analyze the reasons for the second stage. Chen et al. [2] describe the interaction between users and servers. After receiving query information, servers send HTML, scripts and images in different loading phase, respectively, which is possible to use the same TCP connection. As a result, when a connection is established between the client and the server, in order to reduce overhead, the same TCP connection may be reused to request multiple other resources from the server. At this time, the second stage in which only uplink packets

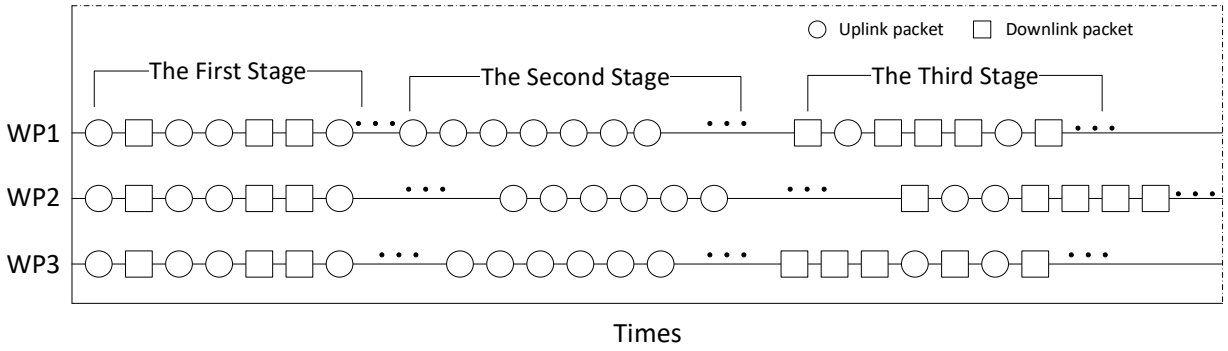

Fig. 3: Typical interactions between the first 100 uplink and downlink packets of three different webpages during the loading process

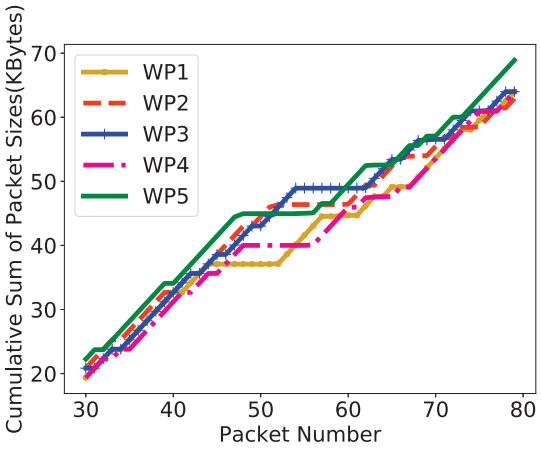

Fig. 4: Cumulative traces of five webpages when set the uplink packet length to 0.

transmitted is appeared. More importantly, for different webpages, the time of the second stage occurs and the number of uplink packets in the second stage are different. This is the key to tackling the webpage fingerprinting barrier.

To simplify the problem, we set the uplink packet length to 0, and still accumulate the first 100 packets. We can expect that the flow trace would become *step-growth* and during the second stage, the flow trace would become smooth. The cumulative traces of five different webpage flows are shown in Fig. 4. As a result, finding the cumulative packet lengths and the number of packets corresponding to the second stage is an effective way to solve webpage fingerprinting problem.

IV. Building Webpage Fingerprinting

In this section, we describe the modeling process for building webpage fingerprinting.

A. Traffic Preprocessing

We first divide the traffic into flows based on a five-tuple representation: (srcIP, dstIP, srcPort, dstPort, protocol (TCP/UDP)), where srcIP represents the client IP address, dstIP represents the server IP address, srcPort represents the client port number, dstPort represents the server port number and protocol represents the communication protocol established between the client and the server. Next, we only save the flows belong to the same website. Since the experimental website is JD (mentioned in III-A), we focus on the Server Hello message in the Service Name Indication (SNI) field, if the "jd.com" string is found in the SNI, the flow will be saved and passed to the next module.

Next, for each webpage, we only consider one flow, which represents the interaction between the client and the server. Those flows that interacting with advertisers or other proxy servers are beyond our consideration.

Finally, we remove those TCP re-transmissions packets because re-transmissions are caused mostly by network conditions.

B. Feature Extraction

To describe the process of webpage loading, we utilize cumulated sum of packet lengths to representation. We designed feature extraction algorithm as exhibited in Alg. 1.

Given the webpage flow $F = (p_1, ..., p_N)$, where $p_i > 0$ represents the downlink packet and $p_i = 0$ represents the uplink packet. We represent the cumulative packet sequence as $A(F) = (a_1, ..., a_N)$, where

$$a_i = \begin{cases} p_i, & \text{if } i = 1; \\ a_{i-1} + p_i, & \text{if } 1 < i \leq N. \end{cases} \tag{2}$$

Next, we divide the cumulative packet length into several intervals

$$R = \{(r_1, r_{1+m}), ..., (r_n, r_{n+m})\} \tag{3}$$

each of which is m in length. Then, we hash these interval into a list

$$I = (v_1, ..., v_n) \tag{4}$$

where

$$v_i = hash(r_i, r_{m+i}) \tag{5}$$

After that, we calculate the number of packets in which the sequence $A(F)$ falling in each interval. Assuming that the

Algorithm 1 $FeatureExtraction$

Input: A flow $F = (p_1, ..., p_N)$ with uplink and downlink packet length and chronological order.
Output: Classification result,that is, the webpage label W corresponding to the flow F

Set the uplink packet length to 0.
Accumulate the packet length to get the accumulated packet sequence $A(F) = (a_1, ..., a_N)$
for each interval $(r_i, r_{i+m}) \in R$ **do**
    $v_i = hash(r_i, r_{i+m})$
    Put v_i into a list $I = (v_1, ..., v_n)$
end for
for each packet $a_j \in A$ **do**
    Calculate v_j corresponding to a_j
    $I(v_j) += 1$
end for
Get the maximum value k_{max} in the list I and its corresponding index v_{max}
Repeat the above procedure until the entire list $A(F)$ is traversed
return Feature set is (v_{max}, k_{max})

interval with the largest number of packets is $(r_i, r_i + 1)$, its corresponding hash value is v_{max}, and the number of packets falling in this interval is k_{max}. (v_{max}, k_{max}) represents the feature set of flow F. We call our webpage fingerprinting approach as *WPF*.

To differentiate the webpage fingerprints, we apply a machine learning technology k-Nearest Neighbor (k-NN) as our classifier. Since the fingerprints have a fixed length and a low dimension, we can directly use them as input to train the k-NN classifier.

V. Performance Evaluation

In this section, we evaluate the performance of WPF, including the evaluation of interval length selection and the comparison with other methods.

A. Preliminary

Performance Measure. The fundamental goal of webpage fingerprinting is to be accurate, i.e., identifying more webpage encrypted flows and avoid misclassification. We consider four metrics to measure the classifiers, namely accuracy, precision, recall and F1-measure. Accuracy is a measure of classification bias, precision is referred to as positive predictive value, recall is referred to as the true positive rate and F1 is the harmonic mean of precision and recall. Before calculating these metrics, we need to define four basic concepts: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN).

Then, precision is calculated according to Eq. (6)

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Recall is calculated according to Eq. (7)

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

TABLE I: Classification accuracy with different n value

n value	2	3	4	5	6	7
Accuracy(%)	91.53	91.58	91.63	91.41	91.33	91.27

F1 is calculated according to Eq. (8)

$$F1 = \frac{Precision * Recall}{2(Precision + Recall)} \tag{8}$$

In our research, the classifier is multi-class classifier. Therefore the accuracy is calculated as the total number of correct classifications divided by the total number of classifications in (9),

$$Accuracy = \frac{TP}{Sum} \tag{9}$$

where Sum means the total number of instances in the classification.

B. Evaluation of Interval Length Selection

In the process of feature extraction, we need to divide the cumulative packet lengths into several intervals, the length of each interval is n. The selection of n is crucial, and the best value of n will change with different flow set. In the experiment, we adjusted the n value several times, for each n value, we conduct 50 times experiments and average the results as experimental results. The classification accuracy is shown in Table I. From Table I we can see, when n is equal to 4, the classification effect is best and the accuracy is up to 91.63%.

C. Comparison With Other Methods

In order to present a comprehensive understanding on the contribution of WPF, we leverage other two methods for comparison.

- *Appscanner* [16] is a robust method that is designed to identify smartphone App using encrypted network traffic analysis. This method also be widely used in website classification. In their work, they use 54 statistics features of uplink flow, downlink flow and complete flow as their feature set. Random forest is used as classifier. Many traffic classification work have achieved high accuracy with this method.
- *DTW* [5] is another website fingerprinting method. In their study, they only use the timing information of upstream packet to identify different websites.

Comparison of Classification Performance. We use the dataset which was described in Section III-A to conduct these comparison experiments. Fig. 5 shows the precision, recall and F1 of classification results using these three methods. From the results we can see that no matter which metrics, WPF performs much better than the other two methods.

Appscanner uses 54 statistics features to represent each flow, however, the flows of webpages on the same website are very similar with each other, thus leading these statistic features lose discrimination. As a result, the precision of classification with Appscanner is only 38.0%. As for DTW,

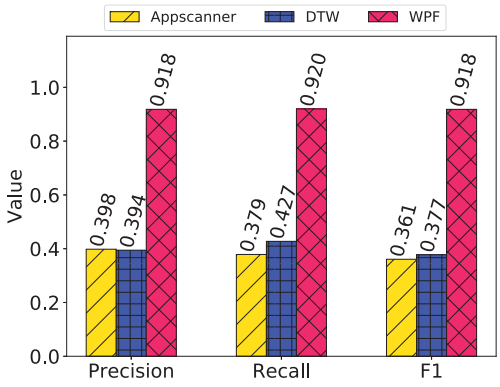

Fig. 5: Classification results with different approaches

TABLE II: Time complexity

Approaches	Extracting Time(s)	Training Time(s)	Testing Time(s)
WPF	1.82E-01	5.95E-05	9.91E-06
Appscanner	5.57E-01	5.87E-03	8.29E-06
DTW	4.91E+03	N/A	2.21E+02

it analyzes each flow and produces 5 prediction results. As long as one of the results is predicted correctly, we determine that DTW prediction is correct. Despite this, the classification precision of DTW is still only 39.4%. In comparison, the classification precision of WPF is much higher than the other two approaches and reaches 91.8%.

Comparison of Time Complexity. To compare time complexity, we test the feature extraction time, training time and testing time of these three methods, respectively, and record them on Table II.

Since the feature extraction and training phase of DTW are the same process, we only record one time. From Table II we can see that no matter which phase, the time complexity of DTW is much higher than the other two approaches. Although DTW only use time information for fingerprinting, timestamps are easily affected by network condition and require warping at first, thus leading to a high time complexity. Comparing with WPF and Appscanner, the extracting time and training time of WPF are less than Appscanner. For testing time, they are close to each other. This result confirms that the time complexity of WPF is lowest in these three fingerprint approaches.

D. Discussion

The above experiments show that WPF is able to effectively classify webpages on the same website. It is superior to other state-of-the-art approaches in terms of classification accuracy and time complexity.

There are two limitations in WPF. The first limitation is that WPF may lose effect if there is no traffic reuse strategy in the loading process of a webpage. To address this problem, we consider to analysis more stages of bidirectional interaction between clients and servers and propose a hierarchical clustering approach. The second limitation is that we do not take into account the traffic of webpages after using proxy. Consequently, it is necessary to expand the traffic dataset and carried out further study.

VI. Conclusion

In this paper, we proposed a simple and efficient fine-grained webpage fingerprinting using only packet length information, where all webpages are from the same website. Based on the features of bidirectional interaction between clients and servers, we set the uplink packet lengths to 0, and abstract the cumulative length of a sequence of packets to represent the fingerprint of a specific webpage. Experimental results show that the accuracy of WPF is up to 91.6% with k-NN algorithm, which is superior than other state-of-the-art fingerprinting approaches. In the future work, we plan to further expand the dataset and consider webpage fingerprinting when using proxy.

References

[1] Jingdong, accessed on may. 5, 2018. [Online]. Available: https://www.jd.com.

[2] Y. Chen, R. Mahajan, B. Sridharan, and Z. Zhang. A provider-side view of web search response time. *acm special interest group on data communication*, 43(4):243–254, 2013.

[3] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions on Information Forensics and Security*, 11(1):114–125, 2016.

[4] T. Dierks and E. Rescorla. Rfc 5246-the transport layer security (tls) protocol version 1.2 (2008). *URL: https://tools.ietf.org/html/rfc5246 (cited on pages 71, 96)*, 2008.

[5] S. Feghhi and D. J. Leith. A web traffic analysis attack using only timing information. *IEEE Transactions on Information Forensics and Security*, 11(8):1747–1759, 2016.

[6] A. Freier, P. Karlton, and P. Kocher. The secure sockets layer (ssl) protocol version 3.0, august 2011. *RFC6101*, 2011.

[7] Y. Fu, J. Liu, X. Li, X. Lu, J. Ming, C. Guan, and H. Xiong. Service usage analysis in mobile messaging apps: A multi-label multi-view perspective. In *IEEE International Conference on Data Mining*, pages 877–882, 2017.

[8] Y. Fu, H. Xiong, X. Lu, J. Yang, and C. Chen. Service usage classification with encrypted internet traffic in mobile messaging apps. *IEEE Transactions on Mobile Computing*, 15(11):2851–2864, 2016.

[9] E. Grolman, A. Finkelstein, R. Puzis, A. Shabtai, G. Celniker, Z. Katzir, and L. Rosenfeld. Transfer learning for user action identication in mobile apps via encrypted trafc analysis. *IEEE Intelligent Systems*, pages 1–1, 2018.

[10] M. Korczynski and A. Duda. Markov chain fingerprinting to classify encrypted traffic. pages 781–789, 2014.

[11] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov. Youtube qoe estimation based on the analysis of encrypted network traffic using machine learning. In *GLOBECOM Workshops*, pages 1–6, 2017.

[12] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel. Website fingerprinting at internet scale. In *Network and Distributed System Security Symposium*, 2016.

[13] G. Ranjan, A. Tongaonkar, and R. Torres. Approximate matching of persistent lexicon using search-engines for classifying mobile app traffic. pages 1–9, 2016.

[14] M. Shen, M. Wei, L. Zhu, and M. Wang. Classification of encrypted traffic with second-order markov chains and application attribute bigrams. *IEEE Transactions on Information Forensics and Security*, 12(8):1830–1843, 2017.

[15] Y. Shi and S. Biswas. Protocol-independent identification of encrypted video traffic sources using traffic analysis. pages 1–6, 2016.

[16] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *IEEE European Symposium on Security and Privacy*, pages 439–454, 2016.