Rethinking Encrypted Traffic Classification: A Multi-Attribute Associated Fingerprint Approach

Yige Chen*†, Tianning Zang*†, Yongzheng Zhang*†, Yuan Zhou‡, Yipeng Wang*†
*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
‡National Computer Network Emergency Response Technical Team/Coordination Center of China

***Abstract*—With the unprecedented prevalence of mobile network applications, cryptographic protocols, such as the Secure Socket Layer/Transport Layer Security (SSL/TLS), are widely used in mobile network applications for communication security. The proven methods for encrypted video stream classification or encrypted protocol detection are unsuitable for the SSL/TLS traffic. Consequently, application-level traffic classification based networking and security services are facing severe challenges in effectiveness. Existing encrypted traffic classification methods exhibit unsatisfying accuracy for applications with similar state characteristics. In this paper, we propose a multiple-attribute-based encrypted traffic classification system named Multi-Attribute Associated Fingerprints (MAAF). We develop MAAF based on the two key insights that the DNS traces generated during the application runtime contain classification guidance information and that the handshake certificates in the encrypted flows can provide classification clues. Apart from the exploitation of key insights, MAAF employs the context of the encrypted traffic to overcome the attribute-lacking problem during the classification. Our experimental results demonstrate that MAAF achieves 98.69% accuracy on the real-world traceset that consists of 16 applications, supports the early prediction, and is robust to the scale of the training traceset. Besides, MAAF is superior to the state-of-the-art methods in terms of both accuracy and robustness.**

***Index Terms*—Encrypted traffic classification, SSL/TLS, domain name, certificate, application data, network management.**

I. Introduction

A. Motivation and Problem Statement

This paper concerns mobile encrypted traffic classification, which is to classify mobile encrypted flows into specific applications. With the rapid development of mobile applications and mobile network in recent years, the number of applications and application downloads in application markets grows continuously [1]. In order to ensure the security of mobile applications, both of the two largest mobile application markets, App Store [2] and Google Play [3], have published enforced encryption standards [4], [5], which adopt Secure Sockets Layer/Transport Layer Security (SSL/TLS) [6]–[8] as their encryption basics. As a result, most of the applications use the encrypted protocol SSL/TLS to communicate with servers, which leads to the high proportion of the SSL/TLS traffic in mobile networks.

High-accuracy encrypted traffic classification is fundamental to plenty of current and future networking and security services, such as policy-based network traffic management, application Quality-of-Service (QoS) and application-level firewalls [9]. Taking application QoS as an example, the enterprise network administrator may intend to prioritize the quality of service for certain specific applications and assign a lower network communication priority to the applications that do not comply with the network policy. The proven encrypted traffic classification approaches focus on identifying encrypted video stream [10] or classifying an encrypted protocol traffic from others [11]. The SSL/TLS oriented approaches [12]–[14] lack a balance between classification accuracy and computational complexity. Therefore, we need to address the crucial problem of encrypted traffic classification to cope with the management of the growing mobile network traffic.

978-1-7281-2700-2/19/$31.00 2019 © IEEE

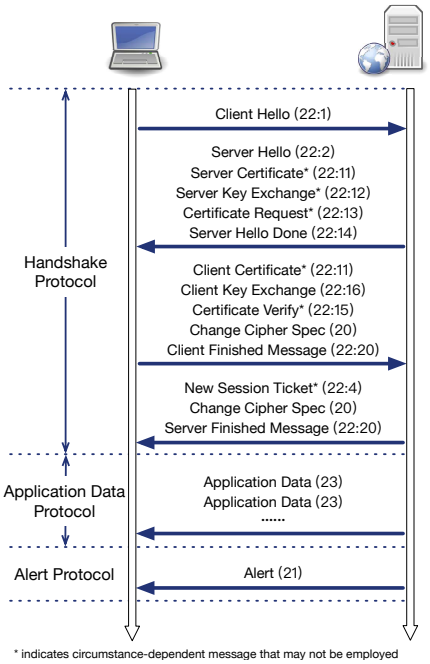


Fig. 1. An Example of the SSL/TLS Protocol Communication Session

B. Related Work and Their Limitations

In this subsection, we first present the basics of the SSL/TLS protocol. Then, we introduce the most relevant and recent encrypted traffic classification methods [12]–[16] and discuss their limitations. Finally, we analyze the limitations of the traceset collection in these relevant papers.

TABLE I
NOTATIONS AND MESSAGE TYPES OF SSL/TLS

Notation	Message Type	Notation	Message Type
20	Change Cipher Spec	22:11	Certificate
21	Alert	22:12	Server Key Exchange
22	Handshake	22:13	Certificate Request
22:0	Hello Request	22:14	Server Hello Done
22:1	Client Hello	22:15	Certificate Verify
22:2	Server Hello	22:16	Client Key Exchange
22:3	Hello Verify Request	22:20	Finished
22:4	New Session Ticket	23	Application Data

1) SSL/TLS Basics: The Secure Sockets Layer (SSL) protocol [6] and its successor Transport Layer Security (TLS) protocol [7], [8] are popular cryptographic protocols that secure the communication between the client and server by encrypting the communication payloads. Figure 1 shows an example of the SSL/TLS protocol communication session. Table I presents the message types and their notations of SSL/TLS. In a typical SSL/TLS session, the two communication sides first exchange `Client Hello` and `Server Hello` to establish the session. Then, the two sides use four kinds of messages, `Server Certificate`, `Server Key Exchange`, `Client Certificate`, and `Client Key Exchange`, to agree on a master secret. The `Change Cipher Spec` and `Server Finished Message` from the server indicates the completion of the handshake. Next, the server starts to transfer `Application Data`, i.e., the communication payload. Finally, the session terminates with an `Alert` message from the server. Due to the high computational cost of public key operations, the protocol allows the server to resume an old session to improve the efficiency of the handshake [7], [8]. Compared with the typical sessions, the resumed session discards `Certificate`, `Server Key Exchange`, and `Certificate Request` while using session IDs or session tickets instead [8].

2) Prior Arts and Their Limitations: The prior arts can be classified into two categories, message-type-based methods, and packet-length-based methods.

The basic idea of the message-type-based methods is to regard the message type sequence of the SSL/TLS session as a Markov chain. Korczyński *et al.* [12] first introduce the concept of the Markov chain fingerprinting and employ first-order homogeneous Markov chains to model possible sequences of the SSL/TLS message types. Shen *et al.* [13], [14] extend the concept of message type Markov chain fingerprinting by incorporating second-order message type Markov chains with the lengths of `Certificate` and first `Application Data` in the session. They propose second-order Markov chain fingerprints with application attribute bigrams (SOB) to make this kind of method more suitable for capturing distinctive characteristics of the applications.

The packet-length-based methods model the packet length sequences through Markov chains or neural networks. Liu *et al.* [15] introduce the concept of Length Block sequence, and propose a method named multi-attribute Markov probability fingerprints (MaMPF). According to the power law distribution of the packet length frequency, MaMPF converts the packet length sequence into a Length Block sequence to reduce the unique number of packet lengths by replacing low-frequency packet length with the closest high-frequency packet length. Then, MaMPF applies first-order Markov chains to model the Length Block sequences. Recently, Liu *et al.* [16] propose Flow Sequence Network (FS-Net), which adopts a multi-layer recurrent neural network based encoder-decoder structure to generate the features of packet length sequence and then directly predict the original applications of the flows.

However, these methods have three limitations. 1) The repeating message type subsequences between different applications grow sharply as the number of applications increases, which greatly weakens the discrimination capability of message types. 2) The methods using attribute bigrams only capture the coarse-grained characteristics and ignore the content of `Certificate`. When the `Certificate` lengths of two different flows are close, their `Certificates` characteristic will be considered the same. As a result, these two flows are likely to be classified as the same application. 3) Some networking and security services need early prediction results to conduct related policies before the transmission of communication payloads. However, these sequence-based methods are unable to accurately predict the original applications of the flows with inadequate sequence information.

3) Limitations of Traceset Collection: The experimental SSL/TLS tracesets in all the mentioned papers are extracted from the unlabeled port mirroring network traces [17] by matching the IP address resolved from the application-related domain names. Indeed, domain names are the preliminary information that typically exists before the establishment of network sessions. However, these tracesets are still not reliable for the following two reasons:

- The application services of the collected tracesets may be incomplete. Mobile applications typically adopt domain names to tag the entrance of certain services. When only partial application-related domain names are used in the traceset collection, the diversity of the application services decreases. In addition, the statistical analysis on our tracesets shows that about 25% - 30% of the flows are not correlated with any domain name and have no chance to be matched in the traceset collection. As a result, the message type sequences or packet length sequences will be more similar in each application but be more distinguishable between different applications, which reduces the difficulty of the classification task and the reliability of the experimental results.
- The application-related domain names employed in the traceset collection may be inappropriate. When different applications use same domain names to direct application services, we are unable to determine the actual application of the matching flows. Whatever application applied to these impure flows, the accuracy of the labels in the traceset will be reduced, which affects the persuasiveness of the experimental results.

This paper addresses all of the above limitations.

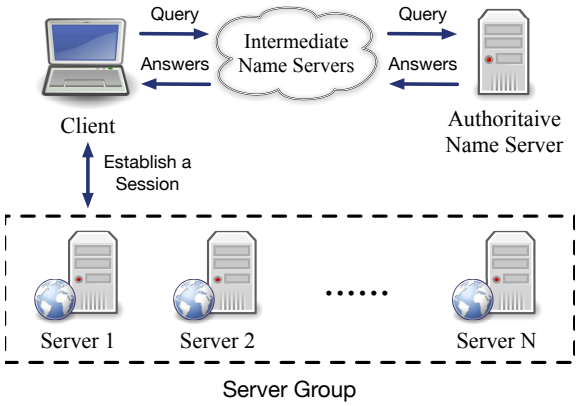


Fig. 2. The Topology of DNS-based Load Balancing

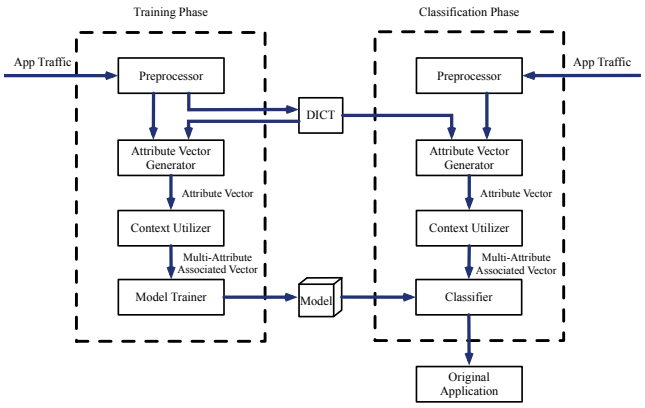


Fig. 3. The System Structure of MAAF

C. Proposed Approach

1) Our Insights into Encrypted Traffic Classification: This paper is based on the following two insights. 1) **The DNS traces generated during the application runtime contain classification guidance information.** The DNS-based load balancing topology, as shown in Figure 2, is popularly employed by applications to balance the workload distribution of the application servers. When an application starts to run, the application client first launches preset domain name queries to obtain the optimal server IP addresses and then establishes sessions with the server for communication. Therefore, the large-traffic domain names are likely to be the application-related domain names. We can use the amount of correlated traffic to represent the correlation between the domain name and the applications in the traceset. 2) **The handshake certificate in the encrypted flows provide clues for the classification.** We notice that some subjects in the X.509 certificates [18] can be essential characteristics for the classification. The subject Common Name [18] (OID = 2.5.4.3, OID is the abbreviation of object identifier) in the end-entity certificate, the final certificate signed for servers [18], typically stores the glob domain name of the servers. The subject Organization [18] (OID = 2.5.4.10) in the end-entity certificate reveals the organization information of the servers. Similar to the domain name mentioned above, we can use the amount of correlated traffic to represent the correlation between the subject and the applications in the traceset.

In order to ascertain the availability of the domain names and certificates, we conduct statistical analysis on the flows starting from a domain name query or containing a certificate in our manually collected traceset. We find that 1) 71.7% of the flows start from a domain name query, 2) 79.9% of the flows contain an X.509 certificate, and 3) 90.7% of the flows either start from a domain name query or contain an X.509 certificate. The statistical results greatly support the availability of the domain names and certificates.

2) Brief Introduction to Our System: Based on our two insights, we propose a multi-attribute-based encrypted traffic classification system named Multi-Attribute Associated Fingerprints (MAAF), which utilizes both the DNS traces generated during application runtime and the certificate in the encrypted flows. Figure 3 presents the system structure of MAAF, which consists of a training phase and a classification phase. The training phase, composed of a preprocessor, an attribute vector generator, a context utilizer, and a model trainer, takes the encrypted flows in the training traceset as input to build the attribute - application correlation dictionaries and train a multi-attribute associated vector classification model. The classification phase, consisting of a preprocessor, an attribute vector generator, a context utilizer, and a classifier, makes use of the dictionaries and classification model from the training phase to predict the original applications of the encrypted flows. The preprocessor in both phases extracts the specified attributes of the encrypted flows, namely Domain Name, Common Name, and Organization, for the attribute vector generation. In particular, the preprocessor in the training phase builds attribute - application correlation dictionaries for the training traceset which store the correlation strength between the attribute values and the applications. Then, the attribute vector generator in both phases generates attribute vectors for the encrypted flows by combining the correlation strength selected from the attribute - application correlation dictionaries with the `Application Data` lengths. Based on the fact that the flows generated by a single application client cluster in the timeline, the context utilizer in both training phase and classification phase exploits the contextual flows to create the multi-attribute associated vectors. The model trainer trains a supervised classification model for the multi-attribute associated vectors of the training encrypted flows. The classifier employs the classification model to classify multi-attribute associated vectors of the unlabeled encrypted flows and predict the applications to which these flows belong.

D. Novelty and Advantages of Our Approach

We conclude the key novelty and advantages of MAAF as follows:

- We excavate the fine-grained characteristics from encrypted flows, namely the correlation strength between the specified attributes and the applications, the `Application Data` lengths, and the contextual information. These characteristics work independently and can be recombined according to the practical needs of networking or security services. For example, we can

discard the `Application Data` lengths to make early predictions with acceptable accuracy. Based on the early prediction results, the networking or security services can conduct the preset policies to the incoming communication payloads of the encrypted flows.
- MAAF overcomes the attribute-lacking problem in the classification caused by the availability of the DNS traces and the certificates in the flows. For the partial-attribute-lacking flows, MAAF can predict their original applications accurately because the specified attributes work independently and can be used individually for accurate prediction. For the full-attribute-lacking flows, we make use of the contextual flows to support the correct classification based on the temporal clustering of the flows generated by an application instance.

E. Key Contributions

We briefly summarize our main contributions as follows:

- We propose Multi-Attribute Associated Fingerprints (MAAF) for the encrypted traffic classification, which exploits DNS traces, certificates, `Application Data` lengths, and contextual flows to predict the original applications of the encrypted flows.
- MAAF constructs attribute vectors for encrypted flows to represent the historical correlation strength between the specified attributes and the applications, and then utilizes the contextual flows to enrich the historical information.
- We evaluate the effectiveness of MAAF by two real-world tracesets. The experimental results show 98.69% accuracy on the manually collected traceset and ideal robustness to different tracesets, which indicates that MAAF outperforms several state-of-the-art methods.

The rest of this paper is organized as follows. Section II introduces the systematic design of MAAF. Section III describes the details of the traceset collection and presents the tracesets used in this paper. Section IV describes the experimental settings and presents the evaluation results. Section V presents the comparison results with the state-of-the-art approaches. Section VI discusses and concludes this paper.

II. Multi-Attribute Associated Fingerprints

In this section, we present the details of the Multi-Attribute Associated Fingerprint (MAAF). MAAF consists of a training phase and a classification phase, as shown in Figure 3.

A. Preprocessor

Both the training phase and the classification phase contain a preprocessor, which extracts Domain Name, Common Name, and Organization from the encrypted flows and constructs attribute - application correlation dictionaries for the training traceset. For the extraction of Domain Name, the preprocessor tries to map the server IP addresses of the flows to the domain names according to the historical DNS traces, that is, searching for the domain names that have been resolved to the server IP addresses. The Common Name and Organization [18] are directly extracted from the end-entity certificates, the final certificate signed to servers as a proof of identity [18], by matching the object identifier 2.5.4.3 and 2.5.4.10, respectively. The end-entity certificates can be found in the handshake of SSL/TLS flows. Considering that the session-resumed flows discard `Certificate` during session handshake, the preprocessor restores the attributes of these flows by referring to their contextual flows. To be specific, for these session-resumed flows, the preprocessor first tries to find the latest historical certificate-containing flow that matches the same host IP addresses and then copies the attribute values of that historical flow.

In the training phase, we also build three attribute - application correlation dictionaries: Domain Name - Application, Common Name - Application, and Organization - Application correlation dictionaries. Each dictionary stores the correlation strength between the attribute values and the applications to support the subsequent modules of both training and classification phases. We use the number of related flows to represent the correlation strength between the attribute value and the application. For instance, suppose we have two applications, namely *Alipay* and *Taobao*. In the network traces of *Alipay*, we find three flows start from Domain Name 'g.alicdn.com' and five flows start from Domain Name 'entpsz.alipay.com'. In the network traces of *Taobao*, we find three flows start from Domain Name 'acs.m.taobao.com' and five flows start from Domain Name 'g.alicdn.com'. So we add the following three correlation entries to the Domain Name - Application correlation dictionary.

$$\begin{cases} (g.alicdn.com) & : (Alipay:3, Taobao:5), \\ (entpsz.alipay.com) & : (Alipay:5, Taobao:0), \\ (acs.m.taobao.com) & : (Alipay:0, Taobao:3). \end{cases}$$

Once we obtain the attribute - application correlation dictionaries, we can get the correlation strength between any attribute value and the applications by querying the correlation dictionaries. When an attribute value does not exist in the dictionary, we consider the correlation strength to be zero. Continuing with the example above, the correlation strength between the Domain Name '`sync.amap.com`' and the applications is

$$(sync.amap.com) : (Alipay:0, Taobao:0).$$

B. Attribute Vector Generator

The attribute vector generator intends to generate the attribute vector for each encrypted flow according to the specified attributes extracted by the preprocessor. For each encrypted flow, the attribute vector generator first searches the specified attributes of the flow in the dictionaries to obtain correlation strength between those attributes and the applications. Then, the attribute vector generator extracts `Application Data` lengths of the encrypted flows and concatenates the correlation strength with the `Application Data` lengths to generate the attribute vectors of the encrypted flows. The optimal number of the `Application Data` lengths R_{app_data} is introduced in Subsection IV-B2 and we suppose R_{app_data} to be three here. Taking an encrypted flow as an example, if the

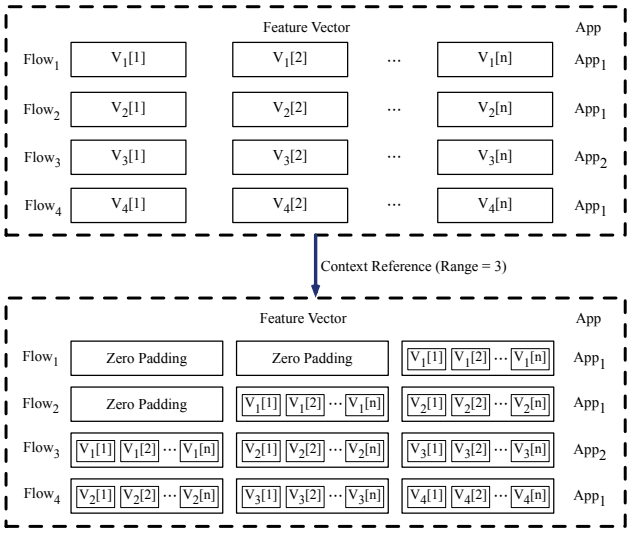


Fig. 4. The Process of Contextual Flow Utilization

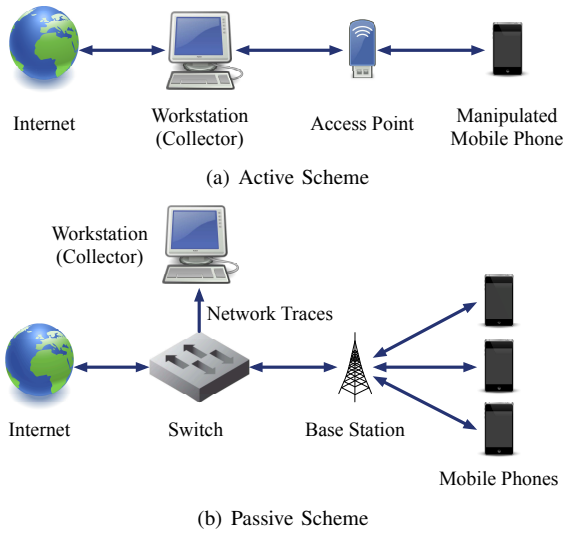


Fig. 5. Mobile Application Traceset Collection Schemes.

correlation strength between its attributes and the applications are

$$\begin{cases} Domain\ Name: & (Alipay:0, Taobao:45), \\ Common\ Name: & (Alipay:2220, Taobao:1955), \\ Organization: & (Alipay:90, Taobao:904). \end{cases}$$

and the lengths of the first three `Application Data` lengths of the encrypted flow are $(272, 773, 0)$, the output attribute vector will be $[0, 45, 2220, 1955, 90, 904, 272, 773, 0]$.

C. Context Utilizer

The context utilizer takes the contextual flows, the latest historical flows before the encrypted flows, into consideration. When a user uses a mobile application, the flows generated by this application client cluster in the timeline, that is, the surrounding flows are very likely to be generated by the same application client. Based on this observation, MAAF can exploit the contextual encrypted flows to reconfirm the classification results.

The process of contextual flow utilization is shown in Figure 4, where we suppose the contextual reference range $R_{context}$ to be three and represent the m^{th} dimension of $\mathrm{Flow_n}$'s attribute vector as $\mathrm{V_n[m]}$. We name the output of the context utilizer as multi-attribute associated vectors. The process of determining the optimal range $R_{context}$ is introduced in Subsection IV-B3. For a given flow, when its contextual flows are sufficient, we exploit the contextual information by directly copying the attribute vectors of the contextual flows in the reversed time order. When the encrypted flow lacks sufficient contextual flows, we fill in the empty position by zero padding to uniform the dimension of multi-attribute associated vectors. For example, in Figure 4, the $\mathrm{Flow_3}$ takes the $\mathrm{Flow_1}$ and $\mathrm{Flow_2}$ as its contextual flows while the $\mathrm{Flow_2}$ can only take the $\mathrm{Flow_1}$ as its contextual flow.

D. Model Trainer and Classifier

We complete the vectorization of the encrypted traffic features after obtaining the multi-attribute associated vectors of the encrypted flows. Next, we apply supervised machine learning models to classify the multi-attribute associated vectors to predict the original applications of the encrypted flows. Considering the weak linear relationship between the specified attributes, we select three non-linear supervised classification models as classifier candidates, namely `C4.5` [19], `Random Forest` [20] and `XGBoost` [21].

These three machine learning models have their strengths. We can choose the appropriate model according to the specific performance requirement. The `C4.5` is a landmark decision tree model that takes the normalized information gain as the splitting criterion. Due to the simplicity of the model, the training and classification speed of `C4.5` is much faster than the other two models, which can be an advantage for massive traffic processing. The `Random Forest` is an ensemble machine learning model that constructs a multitude of decision trees by randomly and repetitively sampling both the training set and features. The sampling mechanism of `Random Forest` makes it suitable for high-dimension vectors and can correct the overfitting habit of the simple decision tree. The `XGBoost` is an implementation of gradient boosting decision tree designed for efficiency and flexibility. Gradient boosting is an approach where new models are created to predict the residuals or errors of prior models and then added together for final prediction [22]. The `XGBoost` can adapt to various data environments and perform high classification accuracy.

According to the comparative evaluation results in Subsection IV-B4, we find that `XGBoost` performs best among the three classifier candidates. Consequently, we select `XGBoost` as the classifier for the multi-attribute associated vectors to achieve the optimal classification accuracy.

III. Traceset Collection

In this section, we first introduce two mobile network trace collection schemes. Then, we provide an overview of our experimental traceset of sixteen popular mobile applications.

TABLE II
THE STATISTIC OF 16 APPLICATION TRACESETS

Developer	Application	Manually Collected Traceset					Automatically Collected Traceset				
		Flows	Packets	Domain	Cert	Both[1]	Flows	Packets	Domain	Cert	Both[1]
Alibaba	Alipay	5201	315234	16.4%	96.3%	97.3%	5929	113902	60.5%	91.4%	93.6%
	Taobao[2]	3231	291348	93.9%	96.8%	99.4%	7766	201895	95.3%	96.9%	100.0%
	AMap[2]	3624	114513	91.7%	98.8%	99.4%	6184	102874	96.7%	98.4%	99.9%
Baidu	Baidu Search	4732	181971	52.5%	90.3%	94.3%	16263	252196	88.2%	97.5%	99.0%
	Baidu Map[2]	5544	215920	40.0%	89.2%	93.8%	25155	663444	54.5%	99.5%	100.0%
Facebook	Facebook	4148	526289	46.3%	82.2%	87.4%	2508	211225	17.8%	66.0%	69.6%
	Instagram	4379	343809	27.0%	5.8%	31.8%	3844	307328	17.5%	42.1%	50.7%
Twitter	Twitter	4463	167166	45.6%	89.7%	93.9%	3638	96616	7.7%	91.2%	92.6%
Sina	Weibo	3817	127057	95.4%	95.2%	99.6%	3558	63036	99.6%	96.2%	100.0%
Airbnb	Airbnb	5843	875837	76.0%	67.7%	82.2%	2329	35278	100.0%	87.8%	100.0%
Linkedin	Linkedin	4203	160614	91.4%	91.8%	98.5%	4267	241124	88.2%	94.5%	99.9%
Evernote	Evernote	7504	202557	98.4%	48.1%	98.5%	822	15036	99.6%	99.1%	99.9%
Blued	Blued	4833	478467	73.4%	55.6%	73.8%	13741	306708	96.4%	95.9%	98.0%
Ele	Ele	6740	99193	98.9%	98.5%	99.9%	8896	148151	98.9%	99.7%	100.0%
Github	Github	4431	151355	98.6%	96.4%	98.8%	1327	50942	97.8%	94.0%	98.5%
Yirendai	Yirendai	4585	61356	98.1%	97.5%	99.2%	6760	64451	97.5%	97.1%	100.0%
Total		77278	4312686	71.7%	79.9%	90.7%	113020	2875337	75.7%	94.4%	96.6%

[1] Both indicates the percentage of flows starting from a domain name query or containing an X.509 certificate.
[2] The applications marked in bold are added to study the ability to classify the same developer's applications.

A. Collection Scheme

Before conducting the evaluation of MAAF, we need to collect mobile application tracesets with ground-truth labels. To the best of our knowledge, there are two schemes to collect tracesets, namely the Active Traceset Collection and the Passive Traceset Collection.

1) Active Traceset Collection: The architecture of this scheme is shown in Figure 5 (a). In this scheme, we collect mobile application tracesets by running applications on a manipulated Android phone and dump the mobile application traces with corresponding labels on the workstation. The Android phone is linked to the workstation via an access point. When the Android phone is running an application, the workstation can capture all the traces generated by this application and label these traces cause the running application is specified by the operator. In general, we can generate mobile application traces by adopting `Monkeyrunner` [23] to fuzz the user interface (UI) of the applications [24], [25] or by employing volunteers to manually operate the applications. Although `Monkeyrunner` can efficiently generate mobile application traces by random UI operations, such as touching and dragging, these generated traces are different from the real traces since the `Monkeyrunner` are unable to interact with applications as the UI designs. The manual mobile application trace generation takes massive human efforts. However, these traces are exactly the real mobile application traces, which guarantees the reliability of the evaluation results.
2) Passive Traceset Collection: The architecture of this scheme is showed in Figure 5 (b). In this scheme, we first collect unlabeled mobile application traces through a port mirroring supporting switch [17] that deployed on the mobile network. In order to label the passive collected mobile application traces, [12]–[15] select labeled encrypted flows from these unlabeled traces by matching the IP addresses resolved from the application-related domain names. Taking Alipay as an example, we know that '*.alipay.com' is an Alipay-related domain name according to its *Whois* [26] record. Then, the *Address* records [27], [28] of this domain name must be the IP addresses of Alipay servers, which can be used to select the encrypted flows of Alipay. However, this scheme faces the problem that the accuracy of the traceset labels will be reduced when several applications share the same application-related domain names.

B. Traceset Introduction

In order to conduct precise experimental evaluations, we collect two different tracesets of sixteen applications through the active collection scheme. The first traceset is manually generated by employing volunteers to operate these applications. The second traceset is automatically generated by adopting `Monkeyrunner` to fuzz the UI of the applications. We refer to the application list in [14] to conduct convincing comparison evaluation with the state-of-the-art approaches. The proportion of encrypted flows in the traces of Netease Music is too small, so we remove it from the list. In addition, we add three applications, namely Taobao, Amap, and Baidu Map, to study the classification of the applications from the same developer. During the trace collection of an application, only this application and the necessary system applications are installed on the Android phone. In the meanwhile, we deploy a firewall to block the noisy traces from Android system applications. The manual traces collection lasts two months while the automatical trace collection only takes two weeks.

Table II presents the statistic of the two tracesets, including the number of flows (Flows), the number of packets (Packets), the percent of flows starting from a domain name query (Domain), the percent of flows containing an X.509 certificate (Cert) and the percent of flows starting from a domain name query or containing an X.509 certificate (Both). The number of the flows in the automatically collected traceset is uneven

because the Monkeyrunner has different adaptability to different applications. According to the statistical results, about 25% - 30% of the encrypted flows are not correlated with any domain name. This means a large number of flows will be omitted by the Passive Traceset Collection scheme. Although there are attribute-lacking flows neither start from a domain query or contain a certificate, our context utilizer can address this challenge and classify these flows correctly.

IV. Evaluation of MAAF

In this section, we conduct rigorous experiments to evaluate the effectiveness of the key modules in MAAF and present the evaluation results in Subsection IV-B.

A. Preliminary

1) Evaluation Schemes: In order to fully understand MAAF, we design the following four comparison experiments on the manually collected traceset to study the effectiveness of MAAF's key modules:

- **Whether to employ Domain Name, Common Name or Organization.** This experiment studies the contribution of each specified attribute and the collaboration between the attributes. We traverse all combinations of the three specified attributes and study the classification accuracy for each combination. Based on the experimental results, we can optimize the combination of the specified attributes to suit the actual network scenario.
- **Different quantities of `Application Data` lengths.** This experiment studies the classification accuracy under different quantities of `Application Data` lengths. Indeed, the `Application Data` carry communication payloads, so we can make a trade-off of using fewer `Application Data` lengths to make earlier predictions. We traverse the quantity of `Application Data` length from zero to nine.
- **Different contextual reference ranges.** This experiment studies the classification accuracy under different contextual reference ranges. The computational cost of MAAF increases as the contextual reference range expands. We can select the optimal contextual reference range by measuring its marginal accuracy return. We traverse the contextual reference range from zero to nine.
- **Different machine learning models.** This experiment studies the adaptability of different machine learning models to the multi-attribute associated vectors. We evaluate the classification accuracy under three machine learning models, i.e., `C4.5`, `Random Forest`, and `XGBoost`.

2) Flow Sequence Shuffle: Considering that the contextual flows in the real-world mobile network may mingle with the flows of other applications, we simulate the real-world flow sequences by shuffling the flow sequences in the traceset. As shown in Figure 9, for each application, we randomly choose some traces in the traceset for the random insertion (In this paper, we roughly choose 10% of the traceset) while keep the relative order of the remaining flows. Then, we randomly insert the selected flows in any position of the original sequence

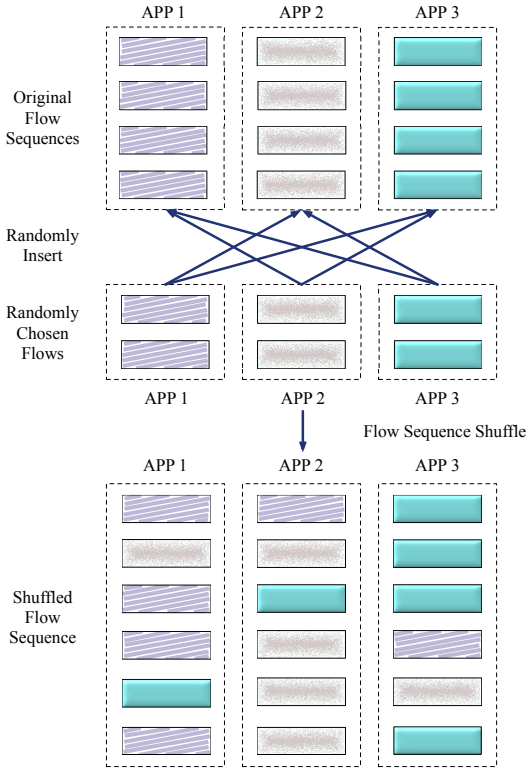


Fig. 9. The Process of Flow Sequence Shuffle

traffic with equal probabilities. Based on the flow sequence shuffle, our experiment can reveal the classification accuracy of MAAF in the real-world mobile network.

3) Cross-Validation: To evaluate the classification accuracy of MAAF, we randomly split the manually collected traceset into a training traceset and a validation traceset. In order to mitigate the impact of traceset partition, we repeat the split ten times and take the average of ten experimental results. In each split, we randomly choose 80% of the flows as the training traceset and take the remaining flows as the validation traceset. For each traceset partition, we perform 5-fold cross-validation on the training traceset to search the optimal hyper-parameters of the classifier, like the max depth of the trees and the number of the estimators. Considering the training time of these models, we conduct 100, 50, 10 random searches for `C4.5`, `Random Forest`, and `XGBoost`, respectively. Then, we employ the optimal hyperparameters to evaluate the classification accuracy on the validation traceset.

4) Criteria of Cross-Validation: We consider Precision (Prec.), Recall (Rec.), Accuracy (Acc), and F1_Macro (F1) as evaluation metrics. For the application App_i, we define the Precision as the rate of the real encrypted flows belonging to App_i in the encrypted flows classified as App_i and the Recall as the rate of the encrypted flows correctly classified as App_i in the total App_i encrypted flows. The F-score of the application App_i is the harmonic average of Precision and Recall. We define Accuracy as the overall rate of all correctly classified encrypted flows in all encrypted flows and F1_Macro as the macro average of all F-score to evaluate the overall classification effectiveness of MAAF.

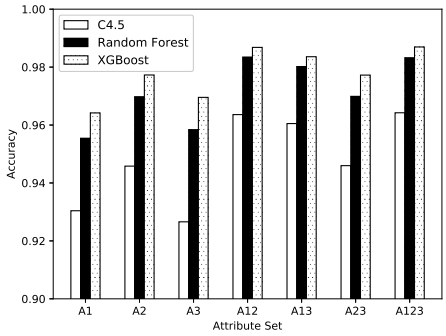


Fig. 6. Classification Accuracy under Different Main Attribute Sets

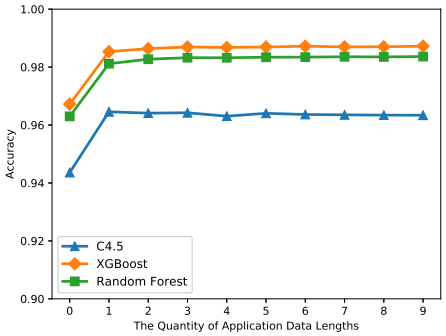


Fig. 7. Classification Accuracy under Different Quantities of `Application Data` Lengths

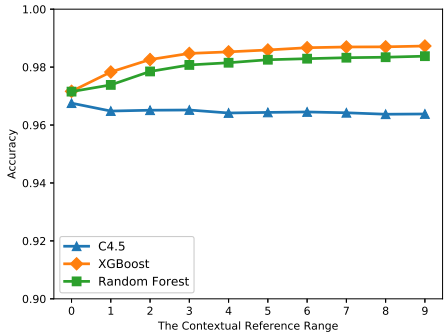


Fig. 8. Classification Accuracy under Different Contextual Reference Ranges

B. Experimental Results of MAAF

1) Whether to Employ Domain Name, Common Name or Organization: The effectiveness of the specified attributes is reflected in Figure 6. Due to the limited space, we replace the attribute names by numbers. We use $A1$, $A2$ and $A3$ to represent Domain Name, Common Name, and Organization, respectively, and use the combination of the numbers to represent multiple attributes. For example, $A12$ denotes that MAAF employs both Domain Name and Common Name.

According to the experimental results in Figure 6, we find that the classification accuracy of MAAF improves with the enrichment of the attributes, which conforms to the ordinary intuition that the classification accuracy improves with the cumulation of valid information. Furthermore, MAAF performs well when only one attribute is employed, which may be attributed to the `Application Data` lengths and the contextual flows. We also notice that the Domain Name and the Common Name can be harmoniously combined to improve the accuracy significantly.

2) Different Quantities of `Application Data` Lengths: Figure 7 shows the upward trend of the classification accuracy with the increase of the quantities of `Application Data` lengths R_{app_data}. The introduction of `Application Data` lengths greatly improves the classification accuracy under all machine learning models. With the subsequent addition of the `Application Data` lengths, the classification accuracy of `Random Forest` and `XGBoost` grows slowly, while the classification accuracy of `C4.5` is slightly reduced. Indeed, the `Application Data` carry the communication payloads. The fewer `Application Data` lengths are used, the sooner MAAF can predict the applications of the encrypted flows. According to the experimental results, we set the quantity of `Application Data` lengths R_{app_data} to three to make a balance between the accuracy and the early prediction.

3) Different Contextual Reference Ranges: In Figure 8, with the increase of the contextual reference range $R_{context}$, the classification accuracy of `Random Forest` and `XGBoost` grows steadily, while the classification accuracy of `C4.5` declines slowly. The dimension of the multi-attribute associated vector linearly expands with the increase of contextual reference range. However, `C4.5` is a simple decision tree method whose ability to select essential features in high-dimension vectors is weaker than the other two models. As a result, it makes sense that the classification accuracy of `C4.5` declines with the increase of the contextual reference range. Based on the experimental results, we conclude that MAAF is computationally cost-efficient when the contextual reference range $R_{context}$ takes around eight.

TABLE III
EXPERIMENTAL RESULTS UNDER DIFFERENT CLASSIFIERS

Application	C4.5		Random Forest		XGBoost	
	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Alipay	0.9630	0.9775	0.9815	0.9898	**0.9859**	**0.9915**
Taobao	0.9293	0.9218	0.9605	0.9623	**0.9671**	**0.9696**
Amap	0.9658	0.9304	0.9816	0.9714	**0.9830**	**0.9767**
Baidu Search	0.8857	0.8579	0.9618	0.9191	**0.9742**	**0.9441**
Baidu Map	0.8807	0.9181	0.9340	0.9717	**0.9517**	**0.9817**
Facebook	0.9685	0.9647	0.9870	0.9815	**0.9891**	**0.9891**
Instagram	0.9703	0.9709	0.9873	0.9835	**0.9922**	**0.9888**
Twitter	0.9761	0.9764	**0.9913**	0.9894	0.9903	**0.9900**
Weibo	0.9840	0.9781	0.9950	**0.9920**	**0.9959**	0.9907
Airbnb	0.9668	0.9696	0.9784	0.9866	**0.9816**	**0.9905**
Linkedin	0.9904	0.9862	0.9955	**0.9975**	**0.9973**	0.9970
Evernote	0.9947	0.9953	**0.9978**	**0.9976**	0.9974	0.9970
Blued	0.9759	0.9818	0.9919	0.9901	**0.9938**	**0.9910**
Ele	0.9839	0.9847	0.9936	**0.9910**	**0.9951**	0.9897
Github	0.9931	0.9916	0.9953	**0.9966**	**0.9962**	0.9964
Yirendai	0.9865	0.9872	0.9946	**0.9957**	**0.9958**	0.9954
Acc/F1	0.9642	0.9627	0.9832	0.9825	**0.9869**	**0.9864**

4) Different Machine Learning Models: The experimental results of different machine learning models are presented in Table III. We notice that `XGBoost` performs best among the three models in terms of Accuracy and F1_Macro. The overall performance of `Random Forest` is better than `C4.5`. For the applications from the same developer, such as *Alipay/Taobao/Amap*, `XGBoost` performs best in both Precision and Recall. For the applications from different developers, `Random Forest` performs better in Recall. This may be caused by the different feature preferences of `XGBoost` and `Random Forest` for high-dimension vectors.

V. COMPARISONS WITH EXISTING APPROACHES

In this section, we compare MAAF with three state-of-the-art approaches in terms of classification accuracy, robustness to the traceset scale, robustness to different tracesets and computational complexity.

A. Preliminary

1) Existing Approaches: (1) SOB [14] integrates the second-order message type Markov chain with bi-gram clus-

tering to classify the encrypted flows. We set the number of bi-gram clustering to 40. (2) MaMPF [15] takes the message type Markov chain and Length Block Markov chain as features and classifies the joint feature vector using `Random forest`. We set Length Block to cover 90% of the whole packets and search the optimal parameters on the training traceset before the validation. (3) FS-Net [16] uses an encoder-decoder structure to generate features and directly classifies the feature vectors by a full-connected neural network. We implement FS-Net using TensorFlow and set the same parameters as the paper.

2) Setting of MAAF: Based on the evaluation results in Section IV-B, we employ all three specified attributes, set the quantities of `Application Data` lengths to three, set the contextual reference range to eight, and take `XGBoost` as the machine learning model. We use the training traceset to perform ten random hyperparameter searches and then use the optimal hyperparameters to achieve ideal accuracy of MAAF.

3) Experimental Setting:

- **Comparisons of Classification Accuracy**: We compare MAAF to the state-of-the-art approaches in terms of the classification accuracy on the manually collected traceset.
- **Comparisons of Robustness to the Scale of Training Traceset**: We compare the classification accuracy of these four approaches when taking 20%, 40%, 60%, 80% of the manually collected traceset as the training traceset and the remaining part as the validation traceset.
- **Comparisons of Robustness to Different Tracesets**: We compare the classification accuracy of the four approaches when training on the automatically collected traceset and validating on the manually collected traceset. We cut the training traceset to balance the size of the tracesets.
- **Evaluation of Computational Complexity**: We deduce and demonstrate the theoretic computational complexity of the preprocessing and classification to compare the throughput of these four approaches.

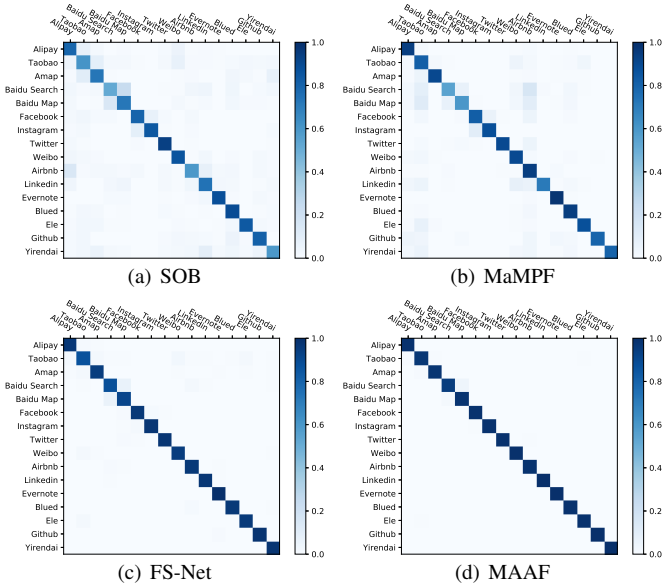
B. Comparisons of Classification Accuracy

The preprocessing of the traceset used in this comparison experiments are the same as Subsection IV-A. Table IV presents the experimental results of different approaches. Compared with the state-of-the-art methods, MAAF performs best in both Accuracy and F1_Macro. The F1_Macro of MAAF is 3.27%, 14.94%, and 23.45% better than FS-Net, MaMPF, and SOB, respectively. In particular, for the applications from the same developer, such as the *Baidu Search*/*Baidu Map* from *Baidu*, MAAF is 9.45%/5.11% better on the Precision, 6.32%/6.66% better on the Recall compared with FS-Net. Compared with MaMPF, MAAF is 15.37%/10.90% better on the Precision, 39.26%/38.07% better on the Recall for the *Baidu Search*/*Baidu Map*. The applications from the same developer is a difficult case to deal with. However, MAAF handles this special case correctly and performs better than the state-of-the-art approaches.

In order to have an intuitive observation of the experimental results, we draw the confusion matrices of the four approaches in Figure 10. SOB and MaMPF misclassify many applications from the same developer, such as *Baidu Search*/*Baidu Map* from *Baidu* and *Facebook*/*Instagram* from *Facebook*. FS-Net significantly ameliorates the misclassification problem except for the *Baidu Search* and *Baidu Map*. MAAF further improves the classification accuracy of these applications, and only few *Baidu Search* and *Baidu Map* flows are misclassified.

TABLE IV
EXPERIMENTAL RESULTS OF DIFFERENT APPROACHES

Application	SOB		MaMPF		FS-Net		MAAF	
	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Alipay	0.7196	0.8062	0.8209	0.9449	0.9707	0.9650	**0.9859**	**0.9915**
Taobao	0.5221	0.6078	0.4739	0.8183	0.8727	0.8718	**0.9671**	**0.9696**
Amap	0.6910	0.7374	0.9321	0.9040	0.9502	0.9484	**0.9830**	**0.9767**
Baidu Search	0.6502	0.4556	0.8205	0.5515	0.8797	0.8809	**0.9742**	**0.9441**
Baidu Map	0.6505	0.7278	0.8427	0.6010	0.9006	0.9151	**0.9517**	**0.9817**
Facebook	0.8319	0.7992	0.8571	0.8205	0.9706	0.9623	**0.9891**	**0.9891**
Instagram	0.9127	0.8445	0.9272	0.8825	0.9819	0.9737	**0.9922**	**0.9888**
Twitter	0.8728	0.9159	0.9703	0.9005	0.9792	0.9729	**0.9903**	**0.9900**
Weibo	0.7807	0.8598	0.7565	0.9020	0.9532	0.9459	**0.9959**	**0.9907**
Airbnb	0.7115	0.5883	0.6598	0.9420	0.9536	0.9646	**0.9816**	**0.9905**
Linkedin	0.6423	0.7575	0.9719	0.7144	0.9648	0.9691	**0.9973**	**0.9970**
Evernote	0.9124	0.8722	0.9486	0.9782	0.9750	**0.9971**	**0.9974**	0.9970
Blued	0.7402	0.8883	0.8508	0.9415	0.9852	0.9645	**0.9938**	**0.9910**
Ele	0.9386	0.8298	0.9569	0.8789	0.9753	0.9606	**0.9951**	**0.9897**
Github	0.7801	0.7925	0.9782	0.7972	0.9871	0.9832	**0.9962**	**0.9964**
Yirendai	0.7464	0.5949	0.9780	0.8043	0.9683	0.9767	**0.9958**	**0.9954**
Acc/F1	0.7601	0.7519	0.8422	0.8370	0.9561	0.9537	**0.9869**	**0.9864**



(a) SOB (b) MaMPF

(c) FS-Net (d) MAAF

Fig. 10. Confusion Matrix of Different Approaches

C. Comparisons of Robustness to the Traceset Scale

Figure 11 presents the classification accuracy of MAAF and three state-of-the-art approaches in different scales of training traceset. As the scale of the training traceset increases, the classification accuracy of SOB and MaMPF increases a lot, which reflects their dependence on the large scale training traceset. FS-Net greatly ameliorates the problem of dependence on the large scale training traceset. MAAF shows strong robustness to the scale of training traceset. We can find that MAAF achieves ideal classification accuracy when it is trained with only 20% of the traceset. According to the experimental results, MAAF shows strong tolerance to the shortage of training traceset, which greatly reduces the difficulty of the traceset collection tasks.

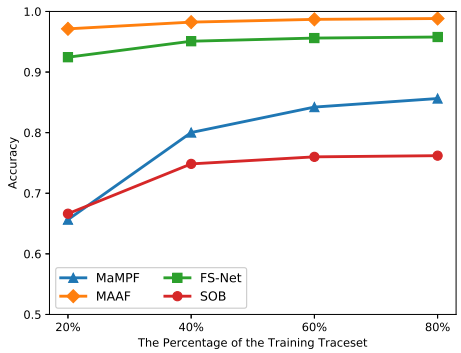


Fig. 11. Classification Accuracy in Different Scale of Training Traceset

D. Comparisons of Robustness to Different Tracesets

Table V presents the experimental results of different approaches when training on the automatically collected traceset and validating on the manually collected traceset. All the state-of-the-art approaches show a great decline on the accuracy and F1_Macro compared with Table IV, which is attributed to the different message type features or different packet length features of the two trecesets. However, the attributes adopted by MAAF are independent of the traceset source, so MAAF still performs well for most of the applications.

E. Evaluation of Computational Complexity

Table VI presents the computational complexity of different approaches. We divide the classification task into preprocessing and prediction. The preprocessing extracts the features from the raw traffic. The prediction handles the features and makes the prediction. We denote the number of the applications, the average number of flows in the application, the average number of messages in the flow, and the average number of packets in the flow as n, f, m, and p, respectively. We deduce the computational complexity as follows:

- SOB traverses all the flows to extract the message type sequences and bi-grams in the preprocessing, so the preprocessing complexity of SOB is $O(nfp)$. Considering that SOB calculates the classification probabilities through n second-order Markov chains, the prediction complexity of SOB is $O(n^2fm)$.
- MaMPF transfers packet lengths to Length Block and calculates the probability features of both message type and length block in the preprocessing, which takes $O(n^2fp)$ computational complexity. In the prediction, MaMPF uses `Random Forest` to classify these probability features. The complexity of `Random Forest` is $O(nfkd)$, where k means the number of trees in the forest and d means the max-depth of these trees.
- The preprocessing of FS-Net traverses all the flows to extracts the packet lengths, which takes $O(nfp)$ computational complexity. The FS-Net contains an encoder, a decoder, and a full-connected neural network. Both the encoder and decoder are constructed of recurrent neural networks. We deduce the prediction complexity of FS-Net to be $O((dpl + hpl + n) * nfh)$, where d means the embedding dimension of packet length, h means the hidden dimension of recurrent neural networks, and l means the number of recurrent neural network layers.
- MAAF extracts the attributes of the flows without the processing of the payload packets, which only need $O(nf)$ computational complexity. The complexity of the attribute vector generator and context utilizer are also $O(nf)$. MAAF employs `XGBoost` as the classifier, whose complexity is the same as `Random Forest`.

For the preprocessing and prediction, we have the relations, $O(nf) < O(nfp) < O(n^2fp)$, and $O(n^2fm) < O(nfkd) < O((dpl + hpl + n) * nfh)$. The computational complexity of MAAF is definitely lower than MaMPF and FS-Net, but slightly higher than SOB.

TABLE V
EXPERIMENTAL RESULTS USING DIFFERENT TRACESETS

Application	SOB		MaMPF		FS-Net		MAAF	
	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Alipay	0.0767	0.6878	0.3733	0.1797	**0.7029**	0.7074	0.6972	**0.9839**
Taobao	0.0555	0.0217	0.2896	0.0544	0.1149	0.2132	**0.9230**	**0.8157**
Amap	0.0270	0.0013	0.4227	0.3201	0.5599	0.3325	**0.9114**	**0.9594**
Baidu Search	0.5762	0.0837	0.1093	0.0798	0.3179	0.2605	**0.7179**	**0.8049**
Baidu Map	0.7291	0.0879	0.2684	0.1933	0.4925	0.4890	**0.8712**	**0.9239**
Facebook	0.7560	0.5836	0.2514	0.5873	0.5735	0.6762	**0.8866**	**0.9482**
Instagram	0.5867	0.8668	0.6346	0.6839	0.6430	0.8879	**0.7362**	**0.9594**
Twitter	0.0179	0.0002	0.2857	0.3640	0.4080	0.8739	**0.8424**	**0.9297**
Weibo	0.1318	0.0095	0.0831	0.0165	0.1433	0.0451	**0.9935**	**0.9768**
Airbnb	0.4784	0.0219	0.9398	0.2013	0.7754	0.3729	**0.9986**	**0.6734**
Linkedin	0.4226	0.0690	0.1113	0.5670	0.6553	0.6315	**0.8118**	**0.9900**
Evernote	0.2194	0.0413	0.8468	0.0985	0.9738	0.3315	**0.9991**	**0.6960**
Blued	0.5342	0.6607	0.0486	0.1129	0.5993	0.6496	**0.9149**	**0.9095**
Ele	0.2896	0.0043	0.0924	0.0401	0.5647	0.2246	**0.9779**	**0.9813**
Github	0.5371	0.3847	0.7389	0.6866	0.3246	0.9549	**0.9212**	**0.9956**
Yirendai	0.2692	0.0053	0.0887	0.0255	0.1703	0.0910	**0.9980**	**0.5685**
Acc/F1	0.2291	0.1962	0.2473	0.2465	0.4775	0.4518	**0.8737**	**0.8738**

TABLE VI
THE COMPUTATIONAL COMPLEXITY OF DIFFERENT APPROACHES

Approach	Preprocessing	Prediction
SOB	$O(nfp)$	$O(n^2fm)$
MaMPF	$O(n^2fp)$	$O(nfkd)$
FS-Net	$O(nfp)$	$O((dpl + hpl + n) * nfh)$
MAAF	$O(nf)$	$O(nfkd)$

VI. DISCUSSION AND CONCLUSION

In this paper, we propose an efficient encrypted traffic classification system Multi-Attribute Associated Fingerprints (MAAF), which makes use of multiple attributes and the contextual flows. The experimental results demonstrate the effectiveness of the attributes and context employed by MAAF and reveal that MAAF performs better than the three state-of-the-art approaches on a real-world traceset. In future work, we plan to study other attributes to improve the classification accuracy of MAAF when applied to different tracesets.

ACKNOWLEDGMENT

The authors would like to thank the shepherd Dr.Nikolaos Laoutaris and the anonymous reviewers for their valuable advices and insightful comments. This work is supported by the National Key Research and Development Program of China under Grant (No.2016YFB0801302, No.2016YFB0801304, No.2016QY05X1002, No.2018YFB0804702, and No. 2018YFB0804704) and Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDC02030100). The corresponding authors are Tianning Zang and Yongzheng Zhang.

REFERENCES

[1] *The State of Mobile in 2019*, https://www.appannie.com/en/insights/market-data/the-state-of-mobile-2019/.

[2] *App Store*, https://www.apple.com/ios/app-store/.

[3] *Google Play*, https://play.google.com/store.

[4] *App Transport Security*, https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CocoaKeys.html#//apple_ref/doc/uid/TP40009251-SW33.

[5] *Network security configuration*, https://developer.android.com/training/articles/security-config.

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

[7] T. Dierks and E. Rescorla, "The transport layer security (tls) protocol version 1.2," 2008.

[8] E. Rescorla, "The transport layer security (tls) protocol version 1.3," 2018.

[9] Y. Wang, X. Yun, Y. Zhang, L. Chen, and T. Zang, "Rethinking robust and accurate application protocol identification," *Computer Networks*, vol. 129, pp. 64–78, 2017.

[10] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1357–1374.

[11] R. Alshammari and A. N. Zincir-Heywood, "Investigating two different approaches for encrypted traffic classification," in *2008 Sixth Annual Conference on Privacy, Security and Trust*. IEEE, 2008, pp. 156–166.

[12] M. Korczyński and A. Duda, "Markov chain fingerprinting to classify encrypted traffic," in *2014 IEEE International Conference on Computer Communications (Infocom)*. IEEE, 2014, pp. 781–789.

[13] M. Shen, M. Wei, L. Zhu, M. Wang, and F. Li, "Certificate-aware encrypted traffic classification using second-order markov chain," in *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE, 2016, pp. 1–10.

[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*, vol. 12, no. 8, pp. 1830–1843, 2017.

[15] C. Liu, Z. Cao, G. Xiong, G. Gou, S.-M. Yiu, and L. He, "Mampf: Encrypted traffic classification based on multi-attribute markov probability fingerprints," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–10.

[16] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *2019 IEEE International Conference on Computer Communications (Infocom)*. IEEE, 2019, pp. 1–9.

[17] J. Zhang and A. Moore, "Traffic trace artifacts due to monitoring via port mirroring," in *2007 Workshop on End-to-End Monitoring Techniques and Services*. IEEE, 2007, pp. 1–8.

[18] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile," 2008.

[19] J. R. Quinlan, *C4.5: programs for machine learning*. Elsevier, 2014.

[20] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 785–794.

[22] J. Ye, J.-H. Chow, J. Chen, and Z. Zheng, "Stochastic gradient boosted distributed decision trees," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 2061–2064.

[23] *Monkeyrunner - Android Developers*, 2019, https://developer.android.com/studio/test/monkeyrunner/.

[24] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," in *2016 IEEE European Symposium on Security and Privacy (EuroS P)*. IEEE, 2016, pp. 439–454.

[25] ——, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2018.

[26] L. Daigle, "Whois protocol specification," 2004.

[27] P. V. Mockapetris, "Domain names: Implementation specification," 1983.

[28] ——, "Domain names: concepts and facilities," 1987.