Encrypted Traffic Classification of Decentralized Applications on Ethereum using Feature Fusion

Meng Shen
Beijing Instititue of Technoloty
Beijing, China
shenmeng@bit.edu.cn

Jinpeng Zhang
Beijing Instititue of Technoloty
Beijing, China
zhangjinpeng@bit.edu.cn

Liehuang Zhu
Beijing Instititue of Technoloty
Beijing, China
liehuangz@bit.edu.cn

Ke Xu
Tsinghua University & BNRist
Beijing, China
xuke@tsinghua.edu.cn

Xiaojiang Du
Temple University
Philadelphia, USA
dxj@ieee.org

Yiting Liu
Beijing Instititue of Technoloty
Beijing, China
liuyiting@bit.edu.cn

ABSTRACT

With the prevalence of blockchain, more and more Decentralized Applications (DApps) are deployed on Ethereum to achieve the goal of communicating without supervision. Users habits may be leaked while these applications adopt SSL/TLS to encrypt their transmission data. Encrypted protocol and the same blockchain platform bring challenges to the traffic classification of DApps. Existing encrypted traffic classification methods suffer from low accuracy in the situation of DApps.

In this paper, we design an efficient method to fuse features of different dimensions for DApp fingerprinting. We firstly analyze the reason why existing methods do not perform well before proposing to merge features of different dimensions. Then we fuse these features by a kernel function and propose a fusion feature selection method to select appropriate features to fuse. Applying features that have been fused to the machine learning algorithm can construct a strong classifier. The experiment results show that the accuracy of our method can reach more than 90%, which performs better than state-of-the-art classification approaches.

ACM Reference Format:
Meng Shen, Jinpeng Zhang, Liehuang Zhu, Ke Xu, Xiaojiang Du, and Yiting Liu. 2019. Encrypted Traffic Classification of Decentralized Applications on Ethereum using Feature Fusion. In *IEEE/ACM International Symposium on Quality of Service (IWQoS '19), June 24–25, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3326285.3329053

1 INTRODUCTION

Ethereum [12] is designed to create an alternative protocol for building Decentralized Applications (DApps). It allows anyone to write smart contract and deploy their DApps. DApps are applications that run on a P2P network of computers, in such way they cannot be controlled by a single entity. By December 2018, there were more than two thousands DApps deployed on Ethereum [7]. From the attack perspective, an adversary can build a fingerprint of a vulnerable application offline, and then uses this fingerprint to identify the target DApp in the open world. The Adversary can also attack specific users, whose habits can be profiled by collecting and analyzing their network activities. From a positive perspective, DApp fingerprinting can help Internet service providers analyze user volume, geographic distribution and habits of DApps, which contributes to the DApp market research. And network manager can optimize network environment according to traffic classification results.

DApps deployed on Ethereum use SSL/TLS protocols to encrypt transmission data between entities. Traditional traffic classification methods (e.g., Deep Packet Inspection) begin to lose effect, due to the adoption of encryption technology. Encrypted traffic fingerprinting is not a new area of research, and indeed many researchers have focused on traffic fingerprinting about websites and smartphone apps. Existing traffic classification methods are based on machine learning algorithms using different traffic statistical characteristics, containing packet sizes [20, 26], packet timestamp series [13], packet flags [17, 24, 25] and packet orderings [26]. It is obvious that network traffic fingerprinting is inextricably bound up with statistical features of traffic from different dimensions.

The complexity of traffic generated by DApps brings challenges to encrypted traffic classification. Traffic features of traditional applications are different because of company-varied implementation SSL/TLS protocol details. The same blockchain platform (Ethereum) is adopted by these applications, which reduces the dissimilarity of traffic generated by different applications. As for classification accuracy, we implement the Markov model [17] generating application fingerprints mainly depending on the flags of SSL/TLS packet, which reaches 56% accuracy. The Random Forest (RF) classifier relying on statistics of packet length [26] achieves the classification accuracy of 80%. Experiment details can be referred to Section 3. As for classification efficiency, Derivative Dynamic Time Warping (DDTW) [13] needs to calculate the warping distance between any two time series, which will cost a lot of time, and is not suitable for the scenario of real-time classification.

In this paper, we propose a novel classification model based on traffic feature fusion. The simplest way of feature fusion is to combine all traffic features of different dimensions together, which has a better classification performance than those classifiers making decisions based on a single dimension features. Not all the features in the feature set have the same importance. By removing the

IWQoS '19, June 24–25, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6778-3/19/06…$15.00
https://doi.org/10.1145/3326285.3329053

influence of relatively bad features, a classifier can make a better classification result than the classifier that makes decisions relying on all dimensional features. Our experiment results show that these two feature fusion strategies only makes non-obvious effects. We propose a feature fusion and feature number increasing method based on a kernel function. By choosing the right number of features to fuse, a high dimensional feature set can be generated. Using the resulting feature set, our classifier outperforms all existing state-of-the-art classification methods in terms of classification accuracy.

The contributions of this paper are as follows:

We make observations on cases where existing classification methods are prone to misclassification on DApps. DApps use SSL/TLS protocols to encrypt their transmission data and adopt Ethereum as their developing platform. The confusion and invisibility of DApp traffic lead to the bad performance of the state-of-the-art encrypted traffic classification method.

We propose a new method to fuse encrypted traffic features of different dimensions. In order to increase the discrimination of DApp fingerprinting, kernel functions can be used to fuse different features and increase feature number based on proper selected features of different dimensions. Feature fusion strengthens the discrimination of DApp's fingerprinting. Different kernel functions, including Polynomial and Radial Basis, are exploited to fuse features. The usage of feature fusion makes great significance to the improvement of classification accuracy.

We verify the effectiveness of the proposed method by experimenting on real traffic datasets collected from 15 representative DApps in different campus sites. These applications are deployed on Ethereum. We collect 18,242 flows and millions of packets in total, and open-source datasets are accessible on Github [8]. Ten-fold cross validation is used to evaluate our method, and results show that our classifier can reach more than 90% accuracy.

The rest of the paper is organized as follows. We introduce the background of DApp and our datasets in Section 2. In Section 3, we analyze the limitation of existing encrypted traffic classification methods and explain the motivation of our feature fusion method. The model is presented in Section 4. Next, we evaluate the proposed method in Section 5 and review the related work in Section 6. Finally, we conclude our paper in Section 7.

2 BACKGROUND AND DATASETS

In this section, we introduce the background of DApp and the datasets we use in our experiments.

2.1 DApp Background

Decentralized applications (DApps) run on a Peer-To-Peer network, which are designed to communicate freely without being controlled by a single entity. DApps are more flexible, transparent, distributed, and resilient with a better incentivized structure software model. Applications deployed on Ethereum are completely open-source, where data and records of application operations are cryptographically stored, using a cryptographic token and generating tokens.

DApps deployed on Ethereum adopt SSL/TLS protocols to encrypt their transmission data, leading to the function loss of deep packet inspection, a traditional traffic classification method. All applications exchange information based on the 443 port number. Thus classifying encrypted traffic of DApps based on the port number is not available, either. Centralized applications are usually managed by different entities. Although they all adopt SSL/TLS protocols, they differ in the details of the protocol implementation. Unlike centralized applications, all DApps are developed on Ethereum, which means that the traffic features of these applications have little difference. State-of-the-art methods of encrypted traffic classification have little effect on the transmission data of DApps. The performance of existing methods is shown in Section 3.

2.2 Datasets

We select 15 DApps of diverse categories on Ethereum, most of which have a user volume more than one thousand per month.

As shown in Figure 1, we deploy our network traffic capture tool in the router of a laboratory. Personal computers are connected to the router, and DApps are installed on the computers. When users use these applications, network traffic is captured and saved into a database. Network flows are then exported to a comma separated value (CSV) file, each row of which contains the information we can get from the packet. The information includes time, source/destination IP addresses, ports, protocols, packet lengths and TCP/IP flags. The traffic of all applications is collected for fifty days. The number of flows and packets for each application is summarized in Table 1. We collect 18242 flows and millions of packets in total from these applications.

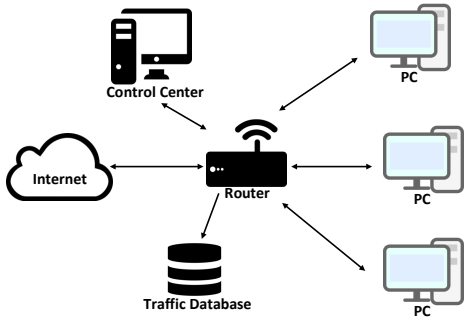

Figure 1: Process of Network Traffic Capture

When using a specific application, a group of packets generated together satisfies the condition that the most recent packets occur within a threshold of time, which are defined as a session. There are several flows in a session. A flow is a sequence of packets, which have the same source/destination IP addresses, sources ports, destination ports and adopt the same transmission protocol. The concept of flow is previously used by Conti et al. [5] and we adopt the same definition here.

3 MOTIVATION

In this section, we firstly analyze existing encrypted traffic classification methods and their performance on our datasets. As these methods on the traffic of DApps do not perform well, we consider merging features of different dimensions to improve the classifier's performance. Then we select those features with a score higher

Table 1: Flow and Packets of Applications

Application	Flow	Packet	Category
Aragon	481	130,483	Governance
Bancor	551	557,057	Trading Token
Canwork	1,342	1,209,033	Social Communication
Chainy	6,768	517,499	Information Protection
Cryptopepes	981	2,995,340	Online Game
Eth_town	548	866,255	Online Game
Etheremon	1,194	149,307	Online Game
Idex	552	6,801,737	Trading Token
Joyso	472	290,622	Exchange Platform
Cryptokitties	697	390,629	Online Game
Lordless	1,423	30,003	Social Contact
Makerdao	511	285,109	Collateralized Dabe
Matchpool	1,438	462,899	Social Communication
Ono	720	803,498	Social Communication
Originprotocol	564	621,997	Online Shopping
Total	18,242	16,111,468	

than 0.15% and merge them together. The results show limited improvement in effectiveness.

3.1 Existing Classification Methods of Encrypted Traffic

In this subsection, we mainly discuss two typical classification methods of encrypted traffic: Markov model relying on SSL/TLS message types and classifier based on statistical features of packet lengths.

Markov Model Relying on SSL/TLS Message Types. Korczynski and Duda propose stochastic fingerprints for application traffic flows conveyed in the Secure Socket Layer/Transport Layer Security (SSL/TLS) session [17, 24]. They use compact notation to represent message types during SSL/TLS sessions (e.g., 22:2 represents `Server Hello`). A sequence of states represents a series of message types sent from server to client. A state is either an SSL/TLS message type (e.g., 22:2) or a sequence of the SSL/TLS message types transmitted in a single TCP segment (e.g., 22:11,22:14). In the Markov chain, they need to calculate the probability of state transition from the previous state to the next state. Besides, they need to calculate the Enter Probability Distribution (ENPD), which represents the probability of the first state of the Markov chain model. Similarly, the Exit Probability Distribution (EXPD) is calculated to represent the probability of the final state of Markov chain model. When an SSL/TLS stream needs to be classified, the result probability is calculated by multiplying ENPD, transition probability and EXPD. The result probability represents how a given encrypted traffic flow is close to a model of an application flow: a larger value means that the encrypted flow is closer to this model.

Based on the datasets we describe in Section 2, we build Markov chain fingerprints for DApps. Although these fingerprints are conductive to the encrypted traffic classification of DApps, they make little effect in some cases. We build a Markov model for Lordless and Bancor. A flow of SSL/TLS message types `22:2,20,22:17 - 23 - 23 - 23 - 23` belongs to Lordless, which we have known its ground turth. We calculate the probability of this flow derived from these two applications. The probability of this flow belonging to Lordless is 0.3648, while the probability of this flow belonging to Bancor is 0.4278. According to the computing result, this flow will be wrongly classified as Bancor, which results in a low precision of Lordless and a low recall of Bancor. Applying our datasets to Mrakov model, the whole accuracy is only 56.65%.

Remark 1: Classifying the encrypted traffic of DApps relying on the SSL/TLS message type is unavailable. The adoption of the same blockchain platform Ethereum causes that the SSL/TLS message types of different applications are similar.

Classifier based on Statistical Features of Packet Lengths. Besides classifying encrypted traffic based on the SSL/TLS message types, Taylor et al. present a automatic fingerprinting based on packet length [26]. They extract 54 statistical features from each flow. These statistical features are calculated from three packet directions: incoming packets only, outgoing packets only and bi-direction packets. For each direction packets, 18 statistical values are calculated: minimum, maximum, mean, median absolute deviation, standard deviation, variance, skew, kurtosis, percentiles (from 10% to 90%) and the number of elements in the series (18 in total). After statistical features are calculated, they adopt Support Vector Classifier (SVC) and Random Forest (RF) to classify network flows of encrypted applications.

We implement this method and the accuracy of the classifier is 79.77% on our datasets. Statistical features of packet length are similar among some applications, for instance, the statistics of Kitty is similar to Originprotocol. For most of these applications, the minimum of incoming packet length is 54, and the maximum of outgoing packet length is 1514. The similarity of statistical features results in the poor performance of the classifier. Obviously, the method based on statistical features of packet length does not apply to our problems.

Remark 2: The method making decisions only depending on packet length information has a limited improvement on classification accuracy. The distinguishability of packet length statistical features is weakened by using the same encrypted protocol and blockchian platform.

3.2 Merging Typical Features for Classification

As mentioned above, the classifier based on the message type of SSL/TLS traffic or the statistical features of packet length does not perform well. Except for the message types and the packet length statistical values, other dimensional features can be used for encrypted traffic classification, including time series and packets bursts. The approach of generating features used in [16] is prevalently exploited in the problem of encrypted traffic fingerprinting. Such set of features has not been previously applied in the encrypted traffic classification of DApps.

Naami et al. [2] propose a feature extraction method that extracts features from consecutive bursts to capture any dependencies that may exist between them. A burst is a sequence of consecutive packets transmitted along the same direction of a TCP network flow. The statistical features of bursts are calculated from burst size and bursts length. Burst size is the number of packets contained in a burst and burst length is the sum of packet lengths in a burst. For the statistical features of burst size or burst length, statistical

values ban be calculated like packet length. Each flow contains bi-directional bursts: ingress bursts and egress bursts and each kind of bursts include burst size and burst length. So we can calculate 72 statistic features from bursts.

As for the statistical features of timing information they are similar to the statistical characteristics of packet length. Three directions of time series are used, and for each series 18 statistical features are calculated (54 features in total).

We use features of three dimensions to different machine learning classifiers separately. Three machine learning classifiers are considered: k-Nearest Neighbor (kNN), Support Vector Machine (SVM) and Random Forest (RF). As for features of packet length, RF performs best and the accuracy is 79.77%. When using statistical features of burst, RF accuracy can reach 82.01%, and neither of the remaining two classifiers works well. If we only take statistical features of time series into consideration, RF outperforms the other two classifiers and the accuracy of this classifier can reach 78.46%. As we can see, the performance of classifier is undesirable, which makes decisions only depending on one dimensional features.

We merge features of three different dimensions together. Except for the 54 statistical values of packet length mentioned in [26], we add the total of packet lengths in each direction. Thus the number of statistical feature calculated from packet length is 57, where $len_1, ..., len_{57}$ represents statistical features of packet length and $bur_1, ..., bur_{72}$ represents the statistical features of burst and $time_1, ..., time_{54}$ represents the statistical features of time series. We merge features of all different dimensions into one dimension vector $merge_1, ..., merge_{183}$. The accuracy of RF reaches 85.50% based on merged features, which performs better than the other two machine learning classification methods. As we can see, the accuracy of merging features has improved 5.73%.

Remark 3: Merging features of different dimensions can improve accuracy of the classifier. The features combined are better than single-dimensional features, though there still exists features with little importance.

3.3 Merging Selected Features for Classification

Not all features in the feature set contribute the same to the result. The curse of diensionality can be avoided by removing the features with little importance. Thus a feature selection function is used to choose the more weighted features. The feature selection function used in this paper relies on the Gini importance metric provided by a Random Forest classifier, which runs on the training set. The percentage contribution of each feature to the classification result is measured. This percentage relies on the Gini impurity index which is computed during estimator building. The classifier gives a score to each feature according to its significance when the training is finished.

We remove features scored lower than 0.15% of different features. And then 166 features are left, which are further exploited to the encrypted traffic classification of DApps. The accuracy varies using different machine learning classification methods based on the selected features. RF performs best compared to kNN and SVM, and its accuracy can reach 85.94%. Compared to the classifier that simply merges features of different dimensions together, RF improves 0.54% accuracy. Although the classification accuracy of this method has been improved, the improvement effect is still unsatisfactory.

Table 2: List of Notations

Notation	Meaning
$F = (p_1, .., p_n)$	Network flow
$PLen$	Packet length of a flow
$PTime$	Time information of a flow
$Burs$	Burst size of a flow
$Burl$	Burst length of a flow
f	Features to fuse
x	One-dimensional features
i	The number of packet length statistical features
j	The number of time series statistical features
k	The number of burst statistical features
VIM	Feature importance
GIm	Gini index

4 FEATURE FUSION AND CLASSIFICATION

In this section, we illustrate our feature fusion method. The whole process of our model is shown in Figure 2. We firstly collect traffic of DApps. Then we extract features of different dimensions. After different dimensional features are extracted, we select the appropriate features to fuse. Using the kernel function, features of different dimensions are fused and the feature number increases at the same time. Finally, we apply the fused features to existing machine learning classification algorithms.

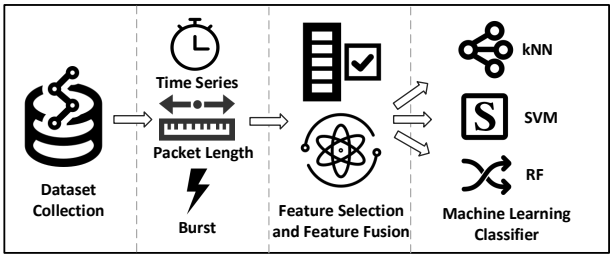

Figure 2: Process of Modeling

4.1 Kernel Functions of Feature Fusion

The notations throughout the paper are summarized in Table 2. A flow of an DApp network traffic can be represented as $F = (p_1, ..., p_n)$, where p_i represents the packet in this flow. For each packet p_i, two different dimensional features can be extracted. They are the packet length information $plen$ and the packet timestamp information $ptis$, where $plen$ represents the bytes that the packet load and $ptis$ represents the timestamp the packet are captured. Thus the packet length and time series can be extracted as follows: $PLen = (plen_1, ..., plen_n)$, $PTime = (ptime_1, ..., ptime_n)$

$$ptime_i = \begin{cases} 0, & \text{if } i = 1; \\ ptis_i - ptis_1, & \text{if } i > 1. \end{cases} \quad (1)$$

In addition to the packet length and time series, we also consider burst behaviors in our situation. A burst is a sequence of consecutive packets transmitted along the same direction of a network flow, and the details of burst actions are illustrated in Figure 3. The blocks in yellow and blue represent different bursts in the figure. For each burst, we extract features from two dimensions: burst size and burst length. Burst size is the number of packets in a burst, and burst length is the accumulative packet length of all packets in that burst. $Burs = (s_1, ..., s_n)$ represents the features of burst size and $Burl = (l_1, ..., l_n)$ represents features of burst length.

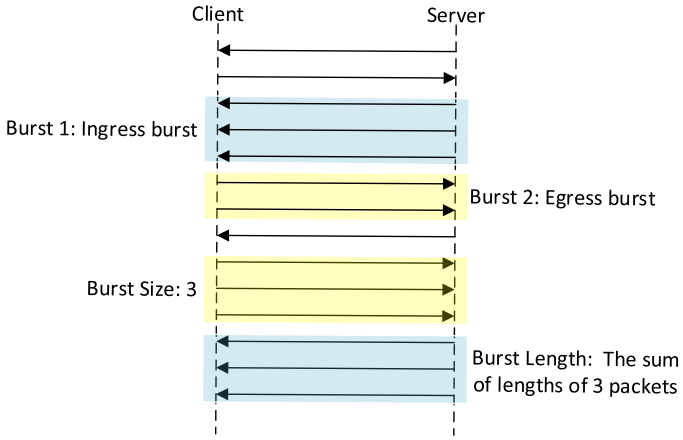

Figure 3: Illustration of Burst Behaviors

Given a decentralization application flow, we firstly need to calculate the statistical features of different dimensions. After statistical features are calculated, we need to select proper features to fuse. The feature selection method will be described in the next subsection. Two kernel functions are used to fuse features of different dimensions: Polynomial Kernel Function and Radial Kernel Function.

Given the features we need to fuse $f = [Plen, Ptime, Burst]$, where $Plen = [[plen_1], ..., [plen_i]]$, $Ptime = [[ptime_1], ..., [ptime_i]]$, and $Burst = [[burst_1], ..., [burst_i]]$, kernel functions are used to feature fusion and feature increase for each kind of dimension. We need to calculate each dimensional fusing result using kernel functions. Then we merge fusion results of all dimensions together.

Suppose x represents the kind of dimensional feature we want to fuse, which can be one of $f = [Plen, Ptime, Burst]$. Firstly, we need to compute the matrix transpose of x, which we marked as x'. x is a matrix of $n * 1$ and x' is a matrix of $1 * n$.

$$x' = x^T \tag{2}$$

The kernel function $K(x, x')$ should be a dot product of characteristic space, which can be presented as $\Phi(x)\Phi(x') = K(x, x')$. The polynomial kernel function is calculated as follows:

$$K(x, x') = (x * x' + 1)^d \tag{3}$$

where $d = 1, 2, ...$ The result of the polynomial kernel function is a matrix of $n * n$.

The radial basis function is another frequently used kernel function, and it is a single-valued function. The Radial Basis kernel function is calculated as follows:

$$K(x, x') = exp(-\frac{\|x - x'\|^2}{2\sigma^2}) \tag{4}$$

where $\sigma \in (0, 1)$. As the growth rate of an exponential function is directly proportional to its value. Exponential function have only one character that the growth rate of such a function is directly proportional to its value. $\|x - x'\|$ is the Euclidean distance. The result of Radial Basis function is also the matrix of $n * n$.

Suppose that the features of different dimensions we need to fuse $Plen$, $Ptime$, and $Burst$ have i, j, k items respectively, the result matrices contain $i * i, j * j, k * k$ items after functions acting on these features. Then we need to merge the features after elevating the feature numbers. The merging method is sequentially appending. The number of features we use to classify varies from $i + j + k$ to $i^2 + j^2 + k^2$. The feature fusing algorithm we design is exhibited in Algorithm 1. The effect of different kernel functions will be discussed in Section 5.

Algorithm 1 Fusing Features Using the Kernel Function

Require: $f = \{Plen, Ptime, Burst\}$
Ensure: $F = \{fea_1, ..., fea_n\}$
for $item \in f$ **do**
  $item' = item.T$
  $v_i = K(item, item')$
  Put v_i into a list $I = (v_1, ..., v_n)$
end for
for $v_i \in I$ **do**
  for each value $v \in v_i$ **do**
    Put v into F
  end for
end for
return

4.2 Fusion Feature Selection

The number of features we use is 183 (57 features of packet lengths, 72 features of bursts and 54 features of time series). Not each feature has the same importance. With the increasing of the feature number in the feature fusing process, the accuracy of the classifier may decline. To find the most suitable fusing features, we can enumerate fusing features' number n from 2 to 183 that achieves the most accurate classification as the model of FFP.

Given a specific value of n, a direct way to evaluate the performance of a classifier is applying the corresponding feature fusing model to the classifier. This simple method would result in high computation overhead when selecting the proper n. Therefore, it is necessary to find an alternative way to efficiently find the best n.

To find the proper n, we mainly consider the importance of different statistic features. We adopt the feature importance measurement method in Random Forest, which calculates the feature importance according to the Gini index. We represent feature importance as VIM. Supposing we have m features, $X_1, X_2, ..., X_C$, we need to calculate the Gini index VIM_j of feature X_j. The Gini index

is calculated as follows:

$$GI_m = 1 - \sum_{K}^{k=1} p_{mk}^2 \tag{5}$$

where K represents the number of applications, and p_{mk} represents the percentage of application k in node m. The importance of feature X_j in node m is calculated as follows:

$$VIM_{jm}^{gini} = GI_m - GI_l - GI_r \tag{6}$$

where GI_l and GI_r represent the Gini index of the two new nodes after node m is branched. When the feature X_j that appears in the node of decision tree i, is included in the aggregate M and its importance in the tree i is calculate according to Equation (7).

$$VIM_{ij}^{gini} = \sum_{m \in M} VIM_{jm}^{gini} \tag{7}$$

Supposing there are n trees in the RF, the Gini index of feature X_j in the trees is calculated as follows:

$$VIM_{j}^{gini} = \sum_{i=1}^{n} VIM_{ij}^{gini} \tag{8}$$

Finally, we need to normalize all the importance scores to get the final importance score. The normalization process is as follows:

$$VIM_j = \frac{VIM_j}{\sum_{i=1}^{c} VIM_i} \tag{9}$$

After the feature importance VIM_j is calculated, we need to sort the feature according the feature importance. Supposing that the larger the feature importance, the greater contribution the feature effects on the classifier, we design a method of measuring cumulative feature contribution with feature number n.

$$CFC_n = \sum_{i=1}^{n} VIM_j \times \frac{1}{n} \tag{10}$$

With the increasing of feature number n, the value of CFC increases. The CFC's increasing speed reflects feature j's contribution to the classifier. By finding the inflection point of CFC curve, we can easily find the optimal fusing feature number.

4.3 Machine Learning Classifier

After kernel-functions are used to fuse features and increase feature numbers, we mainly consider three approaches for using these features to classify encrypted traffic data: k-Nearest Neighbor (kNN), Support Vector Machine (SVM) and Random Forest (RF).

1) k-Nearest Neighbors (kNN): kNN is one of the simplest methods in data mining classification. The core idea is that if a majority of samples in the feature space of the k most adjacent samples belongs to a certain category, the sample has the characteristics of the sample category and belongs to this category. A commonly used distance metric for continuous variables is the Euclidean distance. Although kNN relies on the limit theorem, it is only related to a small number of adjacent samples in decision making.

2) Support Vector Machine (SVM): SVM is a supervised machine learning algorithm that classifies data using labeled training instances falling into different classes [23]. Based on the training data, the classifier fits a hyperplane into the vector space, which can separate the instances into different classes. The plane is as high as possible to make a distinction between different classes. When classifying testing data, the classifier classifies it according to the plan where the testing vector is located.

3) Random Forest (RF): RF refers to a classifier that uses multiple trees to train and predict samples. A RF classifier is an ensemble method that uses weak decision trees to build a strong decision tree. Each decision tree is trained by randomly generated data sets from the original data sets, and the decision results of RF is the decision results of most decision trees. RF is a highly flexible and efficient classifier.

5 PERFORMANCE EVALUATION

In this section, we are dedicated to evaluating the effectiveness of the method we propose. We introduce this section from four subsections: preliminary, evaluation of fusion feature selection, evaluation of classification accuracy and time complexity.

5.1 Preliminary

Methods to Compare. We name the method proposed in this paper as Feature Fusion Fingerprinting (FFP). In order to present a comprehensive understanding on the effects of the method we propose, we leverage two other methods for comparison, which are summarized as follows:

- Markov Model (MARK), which uses SSL/TLS message types to build a Markov model and uses the maximum likelihood function to classify a flow into which kind of web application [17].
- Appscanner (APPS), which uses the statistical features of packet length (e.g., minimum, maximum) about incoming, outgoing, and bidirection flows in the RF classifier [26].

Hardware. The traffic is generated on HP laptops with Intel Core i5-3230M Duo 2.60GHz and 4GB memory, and they are running on Windows 10. Encrypted traffic classification experiments are finished on a server with Intel core duo 3.60GHz and 16GB memory, and it is running on Windows 10.

Cross-validation. For the purpose of validating application fingerprints mentioned in this paper, we use 10 heterogeneous datasets to establish a 10-fold cross validation. We use the 10-fold mean result as our classification result.

Criteria of Cross-validation. The goal of a classifier is to be accurate, i.e., recognize more encrypted flows and avoid misclassification. The criteria used to measure a classifier contain precision, recall, F1-measure and accuracy. Precision is calculated using Equation (11), where TP refers to the number of true positives, FP refers to the number of false positives, FN refers to the number of false negatives, and TN refers to the number of true negatives.

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

Recall is calculated according to Equation (12).

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

F1 is calculated according to Equation (13), and it's the harmonic mean of precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{13}$$

In other words, precision is the fraction of the relevant flows among the detected flows. Recall measures the fraction of a kind of DApp that is detected. And F1 considers both the precision and recall to measure a classifier's score. In our work, the classifier is a multi-class classifier. So the accuracy is calculated as the total number of correct classifications divided by the total number of classifications in (14),

$$Accuracy = \frac{TP}{S} \tag{14}$$

where S represents the total number of instances in the classification.

5.2 Evaluation Fusion Feature Selection

Table 3: Comparison of Different Kernel Function

	FFP_rbf		FFP_poly	
Application	Precision	Recall	Precision	Recall
Cryptopepes	0.8154	0.7976	0.8398	0.8948
Matchpool	0.9422	0.9389	0.9639	0.9526
Lordless	0.9001	0.9281	0.9441	0.9563
Ono	0.7053	0.7524	0.9348	0.9657
Makerdao	0.2473	0.2162	0.5449	0.5834
Canwork	0.7893	0.8468	0.9025	0.8162
Kitty	0.8673	0.7664	0.9076	0.8188
Joyso	0.6703	0.6353	0.7136	0.7636
Eth_town	0.6834	0.2873	0.5854	0.3590
Etheremon	0.9322	0.9058	0.8150	0.9757
Chainy	0.9328	0.9901	0.9788	0.9762
Originprotocol	0.6995	0.7619	0.8702	0.8254
Idex	0.7667	0.4078	0.9063	0.4205
Aragon	0.9743	0.8613	0.9955	0.9152
Bancor	0.6944	0.3425	0.7440	0.4624
F1 Score	0.8897		0.9181	
Accuracy	0.8820		0.9175	

Before been compared to other classification methods, we first evaluate the effectiveness of the fusion feature selection method proposed in Section 4. The evaluation methodology is as follows: we calculate the best n according to CFC_n on part of the dataset and validate n on the rest of the dataset.

We enumerate n from 2 to 183 and calculate the corresponding CFC value defined in Equation (10). The varying trend of CFC is exhibited in Figure 4. We can find that the value of CFC gradually increases with the number of fusing features when $k < 90$. Then it reaches a stage of slow growth during the interval of [90, 120], until finally reaching a stable stage as n continues increasing. Therefore we can choose a number before CFC reaching the stable stage. In this case, n is 120. Next, we enumerate n from 10 to 180 and use fusing features on the classifier. The classification accuracy with the fusing feature number n is shown in Figure 5. With the increase of fusing feature number n, the accuracy of the classifier increases. When n reaches 120, accuracy reaches its highest point. The accuracy declines when n exceeds 120. By combining Figure 4 and Figure 5, we can conclude that when $n = 120$, our method achieves the best classification performance. It also proves the validity of our method to find the most suitable n in Section 4.

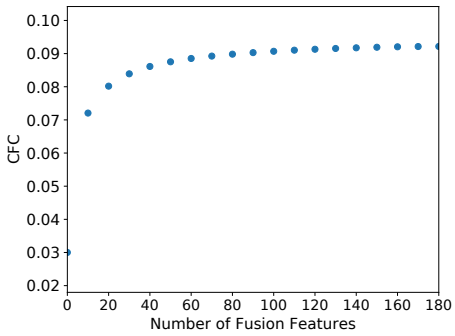

Figure 4: Varying Trend of CFC Value

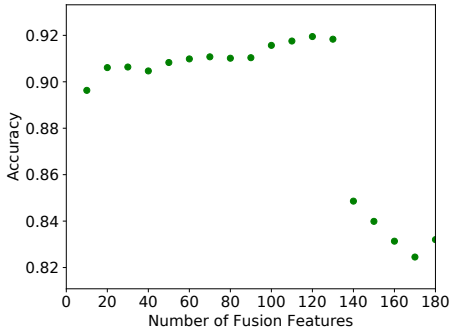

Figure 5: Classification Results for Different Value of n

5.3 Evaluation of Classification Accuracy

In this subsection, we discuss the efficacy of different methods.

Merging features of different dimensions: In Section 3, we present a method that merges features of different dimensions together to make the classifier perform better. We use MFT to represent the method that merges different dimensions' features together and SMT to represent the method that selects features with higher importance to merge together.

Figure 6 shows the F1 score and the accuracy of MFT under three classification algorithms. Whether in F1 score or accuracy, RF performs best. RF accuracy reaches 85.5% higher than kNN and SVM, so we use RF as the classifier of MFT. Table 4 presents the classification results of the four classifiers, among which SMT performs best. The accuracy of SMT reaches 85.94%, which is 0.54% higher than MFT. Compared to Markov, the accuracy increases 29.79% and compared to Appscanner, the accuracy increases 6.17%. The results show that the classifiers perform better by merging features of different dimensions together and removing the features with low importance, which provides a theoretical basis for our feature fusion.

Different kernel functions: Figure 7 shows the F1 score and accuracy of FFP under three classification algorithms. RF performs best compared to other classification methods. So we select RF as

Table 4: Comparison of Markov, Appscanner, MFT and SMF

	Markov		Appscanner		MFT		SMT	
Applications	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Cryptopepes	0.2536	0.3100	0.9991	0.9967	0.7651	0.8596	0.7758	0.8566
Matchpool	0.6943	0.7649	1.0000	1.0000	0.9680	0.9726	0.9705	0.9710
Lordless	0.5667	0.0206	0.9652	0.9969	0.9052	0.9654	0.9109	0.9654
Ono	0.4694	0.2573	0.3182	0.4241	0.9638	0.8904	0.9647	0.8979
Makerdao	0.0000	0.0000	0.0000	0.0000	0.4269	0.3354	0.4759	0.3508
Canwork	0.7199	0.4027	0.9477	0.7831	0.8890	0.8781	0.8879	0.8803
Kitty	0.2476	0.6834	0.1074	0.0179	0.9421	0.8720	0.9425	0.8736
Joyso	0.9112	0.9446	0.9751	0.9430	0.4889	0.7445	0.4839	0.7368
Eth_town	0.0813	0.0765	0.8308	0.6668	0.6827	0.4060	0.6473	0.4107
Etheremon	0.2851	0.3296	0.6367	0.9990	0.9684	0.9641	0.9674	0.9680
Chainy	0.8128	0.7244	0.9552	0.7685	0.9633	0.8186	0.9591	0.8024
Originprotocol	0.1565	0.1802	0.3379	0.9981	0.8106	0.8802	0.8284	0.8802
Idex	0.0509	0.1676	1.0000	1.0000	0.7386	0.5030	0.7024	0.5030
Aragon	0.4582	0.8316	0.8017	0.9936	0.9848	0.9278	0.9827	0.9320
Bancor	0.0279	0.4022	0.9265	0.8409	0.6845	0.5024	0.7088	0.4851
F1 Score	0.5879		0.8080		0.8757		0.8766	
Accuracy	0.5665		0.7977		0.8550		0.8594	

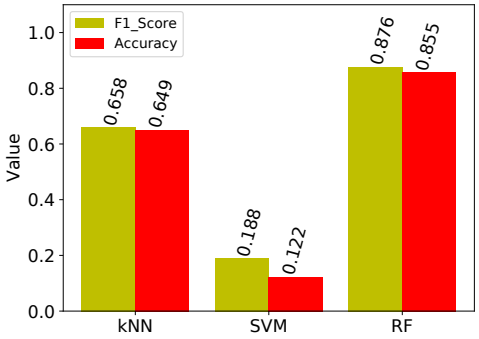

Figure 6: MFT Classification Results with Different Machine Learning Algorithms

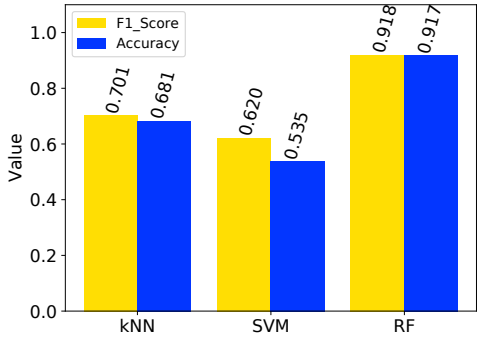

Figure 7: FFP Classification Results with Different Machine Learning Algorithms

our classification algorithm under features using feature fusion. Table 3 presents the classification result of different kernel function. FFP model with a polynomial kernel function can achieve a better classification performance than a radial basis function, whose accuracy reaches 91.75%.

Feature fusion: Figure 8 shows the classification results with different approaches. The accuracy of our method FFP is 91.75%. Compared to Markov, Appscanner, MFT and SMT, the accuracy of our method increases by about 35.10%, 11.98%, 6.25% and 5.81%, respectively. As for F1 score, our method also performs best, which can reach 91.81%. Compared to the other four classification methods, the F1 score of our method increases by about 33.02%, 11.01%, 4.24% and 4.15%, respectively. According to the classification results, our classifier is superior to all the current classifiers.

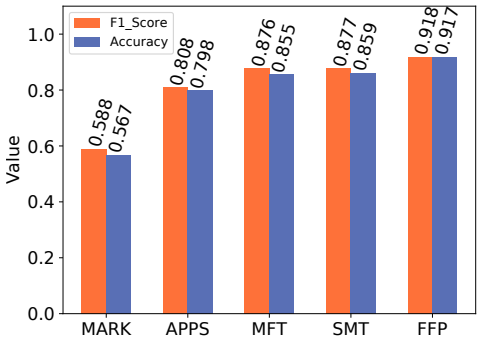

Figure 8: Classification Results with Different Approaches

5.4 Evaluation of Time Cost

In this subsection, we will evaluate the time cost of three classification methods from three aspects: feature extraction cost, training cost and testing cost. Table 5 presents the details of time complexity of different classification methods.

Table 5: Time complexity

Approaches	Feature Extraction(s)	Training Time(s)	Testing Time(s)
Markov	4.084E-3	0.406	2.507E-3
Appscanner	1.226E-2	0.531	2.690E-5
FFP	3.758E-2	30.306	3.943E-3

Feature extraction stage. Three different kinds of classification methods all need to extract features from a network flow. The Markov model needs to extract SSL/TLS packets' types and the transformation relationship between two adjacent packets. As for Appscanner, it only needs to calculate the statistical features of packet length, so it takes the shortest time to extract features. FFP needs to calculate features from three different dimentions and it uses kernel functions to increase feature numbers, so it takes the longest time to extract features. FFP takes 0.03758 second to extract features from a flow, and it takes more 0.02232 second than Appscanner. Mavkov takes the least time to extract features, which only costs 4.0842 milliseconds.

Training and testing stage. As for the training and testing stage, the training time cost is the time cost to fit a classification model. And the testing time cost is an unknown flow to be classified to a specific application. The testing time cost of Markov and Appscanner is similar, which is no more than 0.5 second. Since the input of FFP is higher than Markov and Appscanner, FFP takes the longest time to build a model. When a classification model is constructed, testing time of three classifiers is similar, though FFP needs more time to test a flow. Appscanner takes the shortest time to test a flow. Markov and FFP take the similar time to test a flow. Markov needs to calculate the probabilities of a flow belonging to different applications, and this calculation process takes a larger time cost. FFP's input vectors are much larger than Appscanner, so the testing process of Markov and FFP is longer than Appscanner. Compared to Markov and Appscanner, FFP needs more time to extract features. The vector input into the classifier of PPF is much larger than the classifier of Markov and Appscanner, so the training time and testing time of PPF is longer than Markov and Appscanner.

6 RELATED WORK

6.1 Summary of Encrypted Traffic Classification Methods

With the development of SSL/TLS encrypted protocol, traditional traffic classification methods, such as deep packet inspection [14] and classify traffic based on port number [1], lose their effects. More and more researchers have focused on the classification of encrypted traffic. Besides, some researches have focused on the security protection [9–11, 22, 28, 30] of encrypted traffic. We summarize the encrypted traffic classification methods mainly from the following three aspects:

Web Application Classification. Panchenko et al. proposed a website fingerprinting method at Internet Scale [20]. They characterized the progress of webpage loading with the accumulated sum of packet sizes and used SVM to classify. Cai et al. systematically analyzed existing attacks and defenses to understand which traffic characteristics convey the most information [4]. They proposed a mathematical framework that can evaluate fingerprint attacks and defense performance in the open world. Wang et al. built website fingerprint over Tor [27]. Their work indicated that even clients who use state-of-the-art privacy software (such as the Tor browser) would be snooped by local passive eavesdroppers in real-life conditions.

Mobile Application Classification. Yao et al. focused on the traffic classification problem of smart phone [29]. They built a classifier based on the HTTP header filed and the prefix/suffix surrounding around the HTTP header. Since the HTTP packets are not encrypted, their method lost effect on the encrypted traffic classification. Taylor et al. used 54 statistic features of packet length on RF to build a Appscanner [26]. Conti et al. took the series of packet lengths of each flow and calculated the dynamic warping distance between different flows [5]. In order to reduce the computation burden of the subsequent classification, every kind of action was selected using hierarchical clustering. After warping distances were calculated, they used RF to build a classifier to identify user actions on the phone. They could identify user actions, such as sen and reply messages. Grolman et al. used transfer learning to construct a classifier capable of identifying user actions [15]. They evaluated their method on two different applications Twitter and Facebook. The results showed that their methods were effective on different application versions and devices.

Encrypted Traffic Classification in Other Situations. Encrypted traffic classification could be used in other situations. Barradas et al. used encrypted traffic classification to identify the convert channel in Skype [3]. Their work revealed that previous statements about the unobservability of the convert channel were flawed. Schuster et al. used Convolutional Neural Networks (CNNs) to identify video streams [21]. Since modern video streaming services commonly adopt the MPEG-DASH standard for Dynamic Adaptive Streaming over HTTP. The feature they used was burst size, and they found that burst sizes varied with the content of a video, which mean that a "low action" scene had a low burst size and a "high action" scene had a high burst size. In a video with different video contents, the pattern would have been different.

6.2 Network Traffic of Ethereum

Analysis of Anonymity in Bitcoin. Androulaki et al. used dK-means and Hierarchical Agglomerate Clustering to tie together behavioral patterns based on the dataset that simulates the bitcoin usage in a university setting [19]. They created a list of known bitcoin addresses for each party by actively interacting with parties on the Bitcoin network. In 2014, Koshy et al. applied highly conservative constraints to the Bitcoin network traffic, and they demonstrated that nearly 1,000 bitcoin addresses could be mapped to their specific IPs by leveraging anomalous relaying behaviors [18]. As the traffic was not encrypted, the behavior could be easily discovered by decoding the packet on transmission.

Analysis of Communication Traffic of IoT Devices based on Blockchain Synchronization. Danzi et al. invested several protocols for synchronization between IoT (Internet of Things) devices and the blockchain networks [6]. They proposed a model to

study the impact of the communication link quality and blockchain parameters on the synchronization process. The result showed that the duty cycle of the device should be designed by taking into account both blockchain and communication parameters.

To the best of our knowledge, we are the first one who classify the encrypted traffic generated by DApps. Since the same encrypted protocol SSL/TLS and blockchain platform has been adopted, it is still a serious challenge to classify encrypted traffic generated by DApps. In this paper, we propose an application network traffic fingerprint using feature fusing.

7 CONCLUSION

In this paper, we proposed a feature fusing method based on kernel functions. By generating statistic features from packet lengths, packet bursts and time series using the feature selection function, we could pick out the features that suit for fusing functions. Applying kernel functions on these selected features, we could increase the number of features and fuse them. Then we adopt the fused features on the machine learning algorithm. We conducted experiments to evaluate our method on real traffic datasets collected from 15 representative DApps. The results showed that we proposed method outperforms the other approaches in terms of classification accuracy.

In future work, we plan to further investigate other traffic features of DApps and improve the classification accuracy.

ACKNOWLEDGMENTS

This work is partially supported by the National Key Research and Development Program of China under Grant No. 2018YFB0803405, the National Natural Science Foundation of China under Grant No. 61602039, the Beijing Natural Science Foundation under Grant 4192050, the China National Funds for Distinguished Young Scientists under Grant No. 61825204, the Beijing Outstanding Young Scientist Project, and CCF-Tencent Open Fund WeBank Special Funding.

REFERENCES

[1] Giuseppe Aceto, Alberto Dainotti, Walter De Donato, and Antonio Pescape. 2010. PortLoad: Taking the Best of Two Worlds in Traffic Classification. In *Infocom IEEE Conference on Computer Communications Workshops, 2010*. 1–10.

[2] Khaled Al-Naami, Swarup Chandra, Ahmad M. Mustafa, Latifur Khan, Zhiqiang Lin, Kevin W. Hamlen, and Bhavani M. Thuraisingham. 2016. Adaptive encrypted traffic fingerprinting with bi-directional dependence. In *Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016*. 177–188.

[3] Diogo Barradas, Nuno Santos, and Luís Rodrigues. 2018. Effective Detection of Multimedia Protocol Tunneling using Machine Learning. In *27th USENIX Security Symposium, USENIX Security 2018*. 169–185.

[4] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 227–238. https://doi.org/10.1145/2660267.2660362

[5] Mauro Conti, Luigi Vincenzo Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. 2016. Analyzing Android Encrypted Network Traffic to Identify User Actions. *IEEE Trans. Information Forensics and Security* 11, 1 (2016), 114–125. https://doi.org/10.1109/TIFS.2015.2478741

[6] Pietro Danzi, Anders Ellersgaard Kalør, Cedomir Stefanovic, and Petar Popovski. 2018. Analysis of the Communication Traffic for Blockchain Synchronization of IoT Devices. In *2018 IEEE International Conference on Communications, ICC 2018*. 1–7. https://doi.org/10.1109/ICC.2018.8422485

[7] Dapps. 2019. https://www.stateofthedapps.com/dapps?page=1.

[8] Datasets. 2019. https://github.com/MovanZhang/TrafficDataofDapps/.

[9] Xiaojiang Du and Hsiao-Hwa Chen. 2008. Security in wireless sensor networks. *IEEE Wireless Commun.* 15, 4 (2008), 60–66. https://doi.org/10.1109/MWC.2008.4599222

[10] Xiaojiang Du, Mohsen Guizani, Yang Xiao, and Hsiao-Hwa Chen. 2009. Transactions papers a routing-driven Elliptic Curve Cryptography based key management scheme for Heterogeneous Sensor Networks. *IEEE Trans. Wireless Communications* 8, 3 (2009), 1223–1229. https://doi.org/10.1109/TWC.2009.060598

[11] Xiaojiang Du, Yang Xiao, Mohsen Guizani, and Hsiao-Hwa Chen. 2007. An effective key management scheme for heterogeneous sensor networks. *Ad Hoc Networks* 5, 1 (2007), 24–34. https://doi.org/10.1016/j.adhoc.2006.05.012

[12] Ethereum. 2019. https://www.ethereum.org/.

[13] Saman Feghhi and Douglas J. Leith. 2014. A Web Traffic Analysis Attack Using Only Timing Information. *IEEE Transactions on Information Forensics & Security* 11, 8 (2014), 1747–1759.

[14] Michael Finsterbusch, Chris Richter, Jean Alexander Muller, Klaus Hanssgen, and Eduardo Rocha. 2014. A Survey of Payload-Based Traffic Classification Approaches. *IEEE Communications Surveys & Tutorials* 16, 2 (2014), 1135–1156.

[15] Edita Grolman, Andrey Finkelstein, Rami Puzis, Asaf Shabtai, Gershon Celniker, Ziv Katzir, and Liron Rosenfeld. 2018. Transfer Learning for User Action Identication in Mobile Apps via Encrypted Trafc Analysis. *IEEE Intelligent Systems* 33, 2 (2018), 40–53.

[16] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *25th USENIX Security Symposium, USENIX Security 16*. 1187–1203.

[17] Maciej Korczynski and Andrzej Duda. 2014. Markov chain fingerprinting to classify encrypted traffic. In *2014 IEEE Conference on Computer Communications, INFOCOM 2014*. 781–789. https://doi.org/10.1109/INFOCOM.2014.6848005

[18] Philip Koshy, Diana Koshy, and Patrick D. McDaniel. 2014. An Analysis of Anonymity in Bitcoin Using P2P Network Traffic. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014*. 469–485.

[19] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2013. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 Internet Measurement Conference, IMC 2013*. 127–140.

[20] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website Fingerprinting at Internet Scale. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016*. 1–15.

[21] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. 2017. Beauty and the Burst: Remote Identification of Encrypted Video Streams. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1357–1374.

[22] Meng Shen, Baoli Ma, Liehuang Zhu, Rashid Mijumbi, Xiaojiang Du, and Jiankun Hu. 2018. Cloud-Based Approximate Constrained Shortest Distance Queries Over Encrypted Graphs With Privacy Protection. *IEEE Trans. Information Forensics and Security* 13, 4 (2018), 940–953. https://doi.org/10.1109/TIFS.2017.2774451

[23] Meng Shen, Xiangyun Tang, Liehuang Zhu, Xiaojiang Du, and Mohsen Guizani. 2019. Privacy-Preserving Support Vector Machine Training over Blockchain-Based Encrypted IoT Data in Smart Cities. *IEEE Internet of Things Journal* (2019), 1–1. https://doi.org/10.1109/JIOT.2019.2901840

[24] Meng Shen, Mingwei Wei, Liehuang Zhu, and Mingzhong Wang. 2017. Classification of Encrypted Traffic With Second-Order Markov Chains and Application Attribute Bigrams. *IEEE Trans. Information Forensics and Security* 12, 8 (2017), 1830–1843. https://doi.org/10.1109/TIFS.2017.2692682

[25] Meng Shen, Mingwei Wei, Liehuang Zhu, Mingzhong Wang, and Fuliang Li. 2016. Certificate-aware encrypted traffic classification using Second-Order Markov Chain. In *24th IEEE/ACM International Symposium on Quality of Service, IWQoS 2016*. 1–10. https://doi.org/10.1109/IWQoS.2016.7590451

[26] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2016. AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*. 439–454. https://doi.org/10.1109/EuroSP.2016.40

[27] Tao Wang and Ian Goldberg. 2016. On Realistically Attacking Tor with Website Fingerprinting. *PoPETs* 2016, 4 (2016), 21–36. https://doi.org/10.1515/popets-2016-0027

[28] Yang Xiao, Venkata Krishna Rayi, Bo Sun, Xiaojiang Du, Fei Hu, and Michael Galloway. 2007. A survey of key management schemes in wireless sensor networks. *Computer Communications* 30, 11-12 (2007), 2314–2341. https://doi.org/10.1016/j.comcom.2007.04.009

[29] Hongyi Yao, Gyan Ranjan, Alok Tongaonkar, Yong Liao, and Zhuoqing Morley Mao. 2015. SAMPLES: Self Adaptive Mining of Persistent LExical Snippets for Classifying Mobile Application Traffic. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, MobiCom 2015*. 439–451. https://doi.org/10.1145/2789168.2790097

[30] Liehuang Zhu, Xiangyun Tang, Meng Shen, Xiaojiang Du, and Mohsen Guizani. 2018. Privacy-Preserving DDoS Attack Detection Using Cross-Domain Traffic in Software Defined Networks. *IEEE Journal on Selected Areas in Communications* 36, 3 (2018), 628–643. https://doi.org/10.1109/JSAC.2018.2815442