Exploiting Diversity in Android TLS Implementations for Mobile App Traffic Classification

Satadal Sengupta
IIT Kharagpur, India
satadal.sengupta@iitkgp.ac.in

Niloy Ganguly
IIT Kharagpur, India
niloy@cse.iitkgp.ac.in

Pradipta De
Georgia Southern University, USA
pde@georgiasouthern.edu

Sandip Chakraborty
IIT Kharagpur, India
sandipc@cse.iitkgp.ac.in

ABSTRACT

Network traffic classification is an important tool for network administrators in enabling monitoring and service provisioning. Traditional techniques employed in classifying traffic do not work well for mobile app traffic due to lack of unique signatures. Encryption renders this task even more difficult since packet content is no longer available to parse. More recent techniques based on statistical analysis of parameters such as packet-size and arrival time of packets have shown promise; such techniques have been shown to classify traffic from a small number of applications with a high degree of accuracy. However, we show that when employed to a large number of applications, the performance falls short of satisfactory. In this paper, we propose a novel set of bit-sequence based features which exploit differences in randomness of data generated by different applications. These differences originating due to dissimilarities in encryption implementations by different applications leave footprints on the data generated by them. We validate that these features can differentiate data encrypted with various ciphers (89% accuracy) and key-sizes (83% accuracy). Our evaluation shows that such features can not only differentiate traffic originating from different categories of mobile apps (90% accuracy), but can also classify 175 individual applications with 95% accuracy.

CCS CONCEPTS

• **Networks** → **Packet classification**; Network monitoring; • **Security and privacy** → *Cryptanalysis and other attacks.*

KEYWORDS

traffic classification; randomness; bit-sequence

ACM Reference Format:
Satadal Sengupta, Niloy Ganguly, Pradipta De, and Sandip Chakraborty. 2019. Exploiting Diversity in Android TLS Implementations for Mobile App Traffic Classification. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3308558.3313738

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313738

1 INTRODUCTION

Global mobile data traffic has witnessed astonishing growth over the last half-decade; having multiplied 18-fold since 2011, it is forecasted to account for 20% of all IP traffic by 2021 [1]. The sheer scale and enormous diversity of this traffic has engendered exceptional challenges for network administrators and service providers. One major goal of network administrators is providing differentiated quality of service (QoS) based on the traffic category (also known as service differentiation). The decision of how much bandwidth to provide where, may depend on either preferences of users in the network, or the policies adopted by the enterprise served by the administrator. The first step of such a solution is the classification of traffic originating from various applications. This task, however, has been rendered significantly difficult in case of mobile applications, due to the lack of unique traffic signatures.

Unlike in the case of traditional web traffic, mobile app traffic cannot be segregated based on the presence of obvious signatures (e.g., server address, app port number, unique identifiers in packet content, etc. [12]). In fact, the majority of mobile apps tunnel data over HTTP and HTTPS [24, 28], which use the same port (80 and 443 for HTTP and HTTPS respectively) for all traffic. Also, in case of sub-networks, all traffic is addressed to and from a common IP address (that of the gateway device), rendering IP-based traffic classification infeasible. The most difficult hurdle seems to be the increasing adoption of encryption. Encrypted traffic renders DPI-based efforts to identify signatures from packet content ineffective [26].

One popular way of circumventing this problem is using statistical analysis to classify application traffic [7, 27, 46]. Features such as packet-size, inter-arrival time of packets, etc. have been known to prove beneficial in classifying one type of traffic from another. However, these approaches are under the scanner, and techniques have already been proposed to thwart such attempts. For example, a recent study proposed using packet-size and arrival time obfuscation to confuse a trained classifier [41]. The authors implement their system as a proxy which can fragment and regulate packet departure in a way such that one category of traffic looks exactly like another. In the presence of such techniques, it is necessary to devise strategies which are immune towards obfuscation attempts.

In this paper, we propose a novel set of bit-sequence similarity based features, which leverage footprints left by encryption implementations to achieve traffic classification. A recent study reveals that in case of mobile platforms (and specifically the Android OS), there exists significant diversity in the implementation of TLS/SSL, with a vast majority of those implementations falling

short of the adequate security standards [41]. This phenomenon is mainly due to the *app store* model adopted by major platforms such as Google Android and Apple iOS, which enables developers with little to no expertise to implement security in a host of applications. TLS deployment requires configuring a complicated end-to-end architecture, with numerous 'knobs and buttons'; different permutations of these options adopted by the developers result in a wide array of cipher suites, key-sizes, and certificate management styles getting associated with mobile applications. Additionally, security implementations also depend extensively on the computational capabilities of the mobile devices; while high-end smartphones support the most secure standards, lower-end phones may only support weaker cipher suites, or none at all.

We observe that the differences in encryption implementations of mobile applications in the wild, leave footprints in the encrypted data generated by them, which can be harnessed using a set of novel bit-sequence based features. These features leverage the difference in the randomness of bits, when data is encrypted using different cipher algorithms and key-sizes. We argue that the tell-tale differences in randomness, although weak when data bits are assumed to be in a temporal sequence, begin emerging strongly when the sequence is transformed to the frequency domain (using Fast Fourier Transform). We substantiate this claim with a series of *standard randomness tests*[1] on 11 ciphers, the most noteworthy of which is the *Discrete Fourier Transform* based 'Spectral Test' [36]. This test exhibits the highest standard deviation in randomness scores (~ 0.38) across the ciphers, suggesting its suitability in classifying data generated by different ciphers. We proceed to train a classifier using the aforementioned features, on an offline dataset consisting of data encrypted with 11 different ciphers, and 3 different key-sizes, and obtain 89% (for different ciphers) and 83% (for different key sizes) accuracy in the best case, respectively.

Armed with this set of discriminative features, we propose a mechanism based on machine learning, which can enable classification of mobile network data into source applications. We achieve 71% accuracy in classifying packets from 175 Android apps using only these features, which improves to 95% when augmented with previously defined packet-level features. In comparison, a classifier trained only on baseline features, is able to exhibit only 62% accuracy. Additionally, this technique is not only resistant towards obfuscation, but is also simple to implement, since it requires computing the Fast Fourier Transform of a bit-stream (specialized hardware is available for such tasks [6]), and some additional computation.

The key contributions of this paper include:

(1) We propose and validate a set of novel bit-sequence based features, which enable classification among data encrypted with 11 ciphers and 3 different key-sizes, with 89% and 83% accuracy (best case), respectively (§2).
(2) We substantiate the efficacy of the bit-sequence based features in discriminating among diverse security implementations, by performing multi-class classification of network traffic originating from 175 mobile apps. While previously studied packet-level features (which serve as our baseline) account for only 62% accuracy, our proposed bit-sequence based features alone achieve 71% accuracy, which is boosted significantly (95% in the best case) in conjunction with packet-level features (§3).

[1]Recommended by the National Institute of Standards and Technology (NIST), US Department of Commerce for evaluating suitability of a random bit generator in cryptographic applications [4].

2 BIT-SEQUENCE BASED FEATURES

In this section, we substantiate the notion that data encrypted with different cryptographic system configurations yield different degrees of randomness. We show that it is possible to harness these differences in randomness, and construct "features" (in machine learning parlance) which are useful in extracting sensitive information from encrypted data, e.g., the cipher algorithm or the key-size.

2.1 Background

Secure cryptographic systems are characterized with certain desirable properties, namely (1) the *avalanche effect*, and (2) the *completeness* property. The avalanche effect mandates that a small change (even a bit) in the plaintext should create significant changes (alter a large number of bits) in the ciphertext [25]. The completeness property demands that *all* bits in the plaintext should affect *each* bit in the ciphertext [34]. The broader intent behind these properties is the generation of sufficient randomness in the encrypted data, such that an attacker is unable to exploit any patterns useful in deciphering the plaintext or the encryption key.

However, cipher implementations are seldom perfectly random; existing literature shows that different cipher implementations are characterized with different degrees of randomness [8, 23, 35, 44, 52]. An important measure of the strength of a cipher is therefore the amount of randomness it can induce in the data encrypted with it. While a large number of statistical tests are available to determine randomness, the National Institute of Standards and Technology (NIST), USA recommends a specific set of 15. A candidate cipher is evaluated based on whether the randomness (as determined by these tests) in bits generated due to encryption with it, is sufficiently high to be suitable for cryptographic applications [4]. In this section, we perform these 15 tests on data generated by different ciphers, and validate that the degrees of randomness are indeed different.

2.2 Randomness Analysis of Ciphers

We evaluate the randomness induced by different ciphers on data encrypted by them, by subjecting the data to a set of 15 recommended statistical tests, as mentioned before.

Randomness tests: We perform the following randomness tests: (1) Monobit test, (2) Block frequency test, (3) Independent runs test, (4) Longest runs test, (5) Spectral test, (6) Non-overlapping patterns test, (7) Overlapping patterns test, (8) Universal test, (9) Serial test, (10) Approximate entropy test, (11) Cumulative sums test, (12) Random excursions test, (13) Random excursions variant test, (14) Matrix rank test, and the (15) Linear complexity test. These tests have been defined and standardized in the NIST standard *SP 800-22* [4] [2]. The tests evaluate different properties of the test data which may indicate presence of patterns in the data (e.g., high frequency of specific bit-strings, too many zeros or too many ones, etc.). The score reported for each test is the test statistic *p-value*

[2]The descriptions of these tests have been omitted for the sake of brevity.

Table 1 Results of 15 randomness (recommended by NIST [4]) tests on data encrypted with 11 ciphers. The green row represents the randomness scores for the *Spectral Test*, which is touted as the most powerful test of randomness [36] and also shows the higest std. deviation across tests.

Ciphers/Tests	AES	ARC2	ARC4	Blow-fish	CAST	DES	DES3	PKCS#1 OAEP	PKCS#1 v1.5	XOR	Unenc-rypted	*Standard Deviation*
Monobit Test	0.3563	0.8495	0.4129	0.7034	0.7001	0.6347	0.6617	0.0000	0.0000	0.0000	0.0000	**0.3232**
Block Frequency Test	0.6120	0.6700	0.8508	0.8717	0.6739	0.4795	0.0982	0.7956	0.6285	0.0000	1.0000	**0.2973**
Independent Runs Test	0.2928	0.2092	0.5690	0.9011	0.1275	0.5922	0.0351	0.0000	0.0000	0.0000	0.0000	**0.2950**
Longest Runs Test	0.3735	0.5508	0.2576	0.2018	0.3929	0.5349	0.4236	0.5016	0.2965	0.0000	0.0000	**0.1845**
Spectral Test	0.9641	0.9768	0.7489	0.3135	0.9367	0.5588	0.3628	0.1748	0.9982	0.0000	0.0000	**0.3791**
Non Overlapping Patterns Test	0.3824	0.8874	0.5719	0.7193	0.0479	0.7668	0.1335	0.7917	0.5233	0.0000	0.0000	**0.3263**
Overlapping Patterns Test	0.0123	0.0821	0.2920	0.3004	0.1922	0.8939	0.2518	0.1550	0.1712	0.0000	0.0000	**0.2398**
Universal Test	0.3101	0.2031	0.7855	0.4042	0.8567	0.8574	0.8058	0.3027	0.5298	0.0000	0.0000	**0.3141**
Serial Test	0.2937	0.1751	0.6898	0.8080	0.3829	0.6550	0.0151	0.5689	0.0489	0.0000	0.0000	**0.2931**
Approximate Entropy Test	0.3202	0.0553	0.5882	0.5285	0.2862	0.0851	0.3340	0.2450	0.1651	0.0000	0.0000	**0.1909**
Cumulative Sums Test	0.3824	0.9320	0.5635	0.6605	0.8950	0.3800	0.6888	0.0000	0.0000	0.0000	0.0000	**0.3498**
Random Excursions Test	0.8251	0.4261	0.4576	0.9930	0.8197	0.3436	0.4022	0.2735	0.5566	0.7000	0.7000	**0.2208**
Random Excursions Variant Test	0.6991	0.8571	0.3518	0.2654	0.8173	0.2099	0.3670	0.0811	0.4618	1.0000	0.3865	**0.2839**
Matrix Rank Test	0.5624	0.9083	0.5359	0.9268	0.7473	0.5678	0.3579	0.1282	0.8707	0.0000	0.0000	**0.3323**
Linear Complexity Test	0.9053	0.8132	0.8263	0.2563	0.7104	0.0709	0.8156	0.2682	0.8527	0.2072	0.3292	**0.3047**

which is a function of the observed (encrypted) data. The *p-value* indicates the probability that a perfect random number generator would have produced a sequence (in terms of the property assessed by a particular test) less random than the tested sequence. If *p-value* = 1 for a test then the tested sequence is deemed to have perfect randomness, while if *p-value* = 0 then the sequence may be adjudged as completely non-random.

Ciphers: As candidates for the randomness tests, we consider 7 symmetric-key ciphers (e.g., AES, ARC2, ARC4, Blowfish, CAST, DES, and DES3), 2 asymmetric ciphers (e.g., PKCS#1 OAEP and PKCS#1 v1.5), and the XOR cipher. The variety in types of ciphers ensures generalizability. We also consider unencrypted data as the 11^{th} category for these tests (we refer to this as a cipher here onward for ease of explanation).6 The implementations of the considered ciphers are as available in the Python library `PyCrypto` [3].

Setup: We synthesize random data blocks of 1504 bytes each, which is approximately the MTU (maximum transmission unit) of a network packet. We use the `random.SystemRandom` class provided by Python for this purpose. We generate and encrypt 1000 such data blocks (or packets) for each cipher mentioned above. In case of the symmetric ciphers, we use a fixed key sized 128 bits. We use the Cipher-Block Chaining (CBC) encryption mode in all cases (of symmetric ciphers). For the asymmetric ones, we generate a 1024 bit RSA public/private key pair using `OpenSSL`, and then extract the public key; this public key is then used for encryption. The encrypted data generated by each cipher is converted into a separate bit-stream and fed as input to the randomness testers. We implement the randomness testers based on the Python library `r4nd0m` provided by NIST [2].

Results: The results of the randomness tests are provided in Table 1. We report scores, which are *p-values* (as mentioned before), with a precision of only up to 4 places after the decimal point. Therefore, a score of 0.0000 may not indicate a *p-value* of exactly zero, but a value which is lesser than 0.00005 (since we round up). Most tests find unencrypted data and data encrypted with XOR less than satisfactory (very low *p-value*) for usage in practical cryptographic systems, as expected. More importantly, we observe that there are discernible differences in the randomness values for data generated by the 11 ciphers considered, with respect to most of the tests. It is meaningful to ask therefore: *which test should one focus on if she intends to classify data based on the cipher used to encrypt it?* We attempt to answer this question next.

Towards identification of classification features: Of the 15 tests considered in the aforementioned experiment, the test known as the ***Spectral Test*** (also called the *Discrete Fourier Transform (DFT) Test*) is of special significance both in the literature and in this work. The roots of this test lie in a work by Coveyou et al. which suggested employing Fourier analysis to analyze the quality of random number generators [19]. Donald Knuth formalized the *Spectral Test* based on this idea and touted the test as "by far the most powerful test known", since certain weak ciphers which pass all other tests have been known to invariably fail this particular test [36]. We also observe that this test (highlighted with green in Table 1) shows the highest standard deviation in scores across all ciphers, which is indicative of high variability. Buoyed by these observations, we choose the DFT-based *Spectral Test* as the basis of defining classification features for traffic encrypted with different encryption parameters.

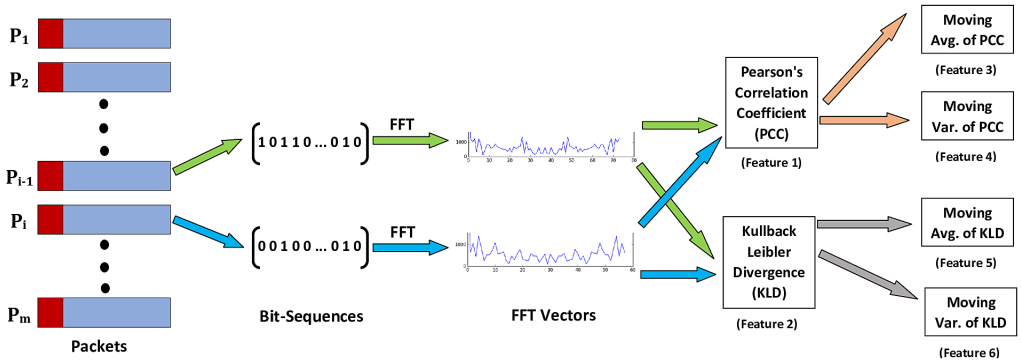
2.3 Defining Bit-Sequence Based Features

Having zeroed in on the basis of feature extraction, i.e., the DFT-based *Spectral Test*, we proceed to examine how we can define features which can effectively classify encrypted data based on encryption parameters.

Role of Fast Fourier Transform (FFT): FFT is an efficient algorithm for computing the DFT of a signal (or sequence). FFT transforms the input sequence (a bit sequence in our case) from the temporal domain to the frequency domain; the result is a vector containing the intensities of each frequency component in the original sequence. Based on our observations in the previous subsection, we already know that analyzing the FFT vectors generated from encrypted data (as done in the *Spectral Test*) is beneficial in gleaning out the randomness present in encrypted data. However, *how does one use FFT effectively to leverage this property and subsequently enable classification?*

The case for packet pairs: Randomness is likely to be induced in an individual app packet from two sources: (1) the content encapsulated in the packet, and (2) encryption. Our aim in this work is to capture the randomness due to encryption only. This requirement drives us to consider a pair of packets instead of individual packets; the idea is to eliminate the randomness due to differences in content,

Figure 1 Feature Extraction: Pictorial description of the methodology used to extract the 6 defined bit-sequence based features



as far as possible. Content similarity is likely to be the maximum in *consecutive packets* generated by an application, rather than any random pair of packets. For example, in the case of a video app, consecutive packets are likely to carry two consecutive frames of a video. Therefore, in this work, we consider consecutive packets for defining our features. We discuss possible scenarios arising out of this requirement from an implementation perspective, and ways to address those, in § 3.2.

Bit-sequence based features: We posit that the randomness due to encryption can be best leveraged by comparing the FFTs of two consecutive packets from a source app. In other words, we determine how similar or dissimilar the FFT vectors of two consecutive app packets are. The similarity (or lack thereof) in randomness is captured using 2 standard metrics employed in measuring the similarity between 2 *distributions*: (1) Pearson's Correlation Coefficient (PCC), and (2) Kullback-Leibler Divergence (KLD). In our case, these *distributions* are the Fourier transformed bit-sequences of encrypted packets generated by a cryptosystem.

In terms of implementation, we first convert the bit-sequence of each network packet into its corresponding bit-sequence in the frequency domain, by performing FFT. Thereafter, we compute the PCC and KLD of such bit-sequences of 2 consecutive packets. In order to not get restricted by sudden changes in PCC and KLD values, we also consider the moving average and moving variance of these values, which are smoother and more consistent. The entire mechanism is illustrated in Fig. 1.

Specifically, we define the following features to quantify bit-sequence similarity (or difference) between packets originating from a cryptosystem:

(1) *Pearson's Correlation Coefficient (PCC)* between the Fast Fourier Transform (FFT) of bit-sequences of 2 consecutive packets.
(2) *Pearson's Correlation Coefficient Average (PCCA)*, which is the moving average of the PCC values obtained.
(3) *Pearson's Correlation Coefficient Variance (PCCV)*, which is the moving variance of the PCC values obtained.
(4) *Kullback-Leibler Divergence (KLD)* between the FFTs of bit-sequences of 2 consecutive packets.
(5) *Kullback-Leibler Divergence Average (KLDA)*, which is the moving average of the KLD values obtained.
(6) *Kullback-Leibler Divergence Variance (KLDV)*, which is the moving variance of the KLD values obtained.

Table 2 Classification results (precision ($\mathcal{P}$), recall ($\mathcal{R}$), f1-score ($\mathcal{F}$), accuracy ($\mathcal{A}$)) obtained using 3 different classifiers for 3 variations of packet-sizes, for packets originating from 11 sources (10 encrypted using different ciphers, and 1 unencrypted). Best classification performance is achieved using the Random Forest Classifier (highlighted in green).

Classifier / Packet-size	**All Equal**	**Equal for Source**	**All Different**
Metrics	$\mathcal{P}, \mathcal{R}, \mathcal{F}, \mathcal{A}$	$\mathcal{P}, \mathcal{R}, \mathcal{F}, \mathcal{A}$	$\mathcal{P}, \mathcal{R}, \mathcal{F}, \mathcal{A}$
Random Forest	0.89, 0.89, 0.89, **0.89**	0.96, 0.96, 0.96, **0.96**	0.90, 0.89, 0.89, **0.89**
Decision Tree	0.88, 0.87, 0.87, **0.87**	0.93, 0.93, 0.93, **0.93**	0.87, 0.86, 0.86, **0.86**
K Nearest Neighbors	0.81, 0.81, 0.81, **0.81**	0.93, 0.93, 0.93, **0.93**	0.72, 0.72, 0.72, **0.72**
Gaussian Naive Bayes	0.73, 0.61, 0.60, **0.61**	0.88, 0.84, 0.82, **0.84**	0.75, 0.71, 0.70, **0.71**
M[1]layer Perceptron (Deep)	0.65, 0.69, 0.66, **0.69**	0.89, 0.89, 0.89, **0.89**	0.57, 0.61, 0.58, **0.61**
Logistic Regression	0.58, 0.62, 0.57, **0.62**	0.69, 0.66, 0.63, **0.66**	0.53, 0.53, 0.50, **0.53**
Support Vector Machine	0.50, 0.46, 0.41, **0.46**	0.77, 0.71, 0.66, **0.71**	0.46, 0.44, 0.39, **0.44**
AdaBoost	0.24, 0.28, 0.21, **0.28**	0.36, 0.29, 0.22, **0.29**	0.34, 0.36, 0.31, **0.36**

Let us denote the FFT vectors corresponding to the n^{th} packet and $(n-1)^{th}$ packet as p^n and p^{n-1} respectively. Pearson's correlation coefficient (PCC) between the FFTs corresponding to the n^{th} and the $(n-1)^{th}$ packets is defined as:

$$PCC_n = \frac{\sum_i (p_i^n - \overline{p^n}).(p_i^{n-1} - \overline{p^{n-1}})}{\sqrt{\sum_i (p_i^n - \overline{p^n})^2 . \sum_i (p_i^{n-1} - \overline{p^{n-1}})^2}} \quad (1)$$

The features derived from PCC, i.e. PCCA and PCCV, corresponding to the n^{th} packet in sequence, are defined as follows:

$$PCCA_n = \frac{1}{n}\sum_{i=0}^{i=n} PCC_i \quad (2) \qquad PCCV_n = \frac{1}{n}\sum_{i=0}^{i=n} (PCC_i - PCCA_i)^2 \quad (3)$$

Kullback-Leibler divergence between the FFTs corresponding to the n^{th} and the $(n-1)^{th}$ packets, is defined as follows:

$$KLD_n = \sum_i p_i^n . log \frac{p_i^n}{p_i^{n-1}} \quad (4)$$

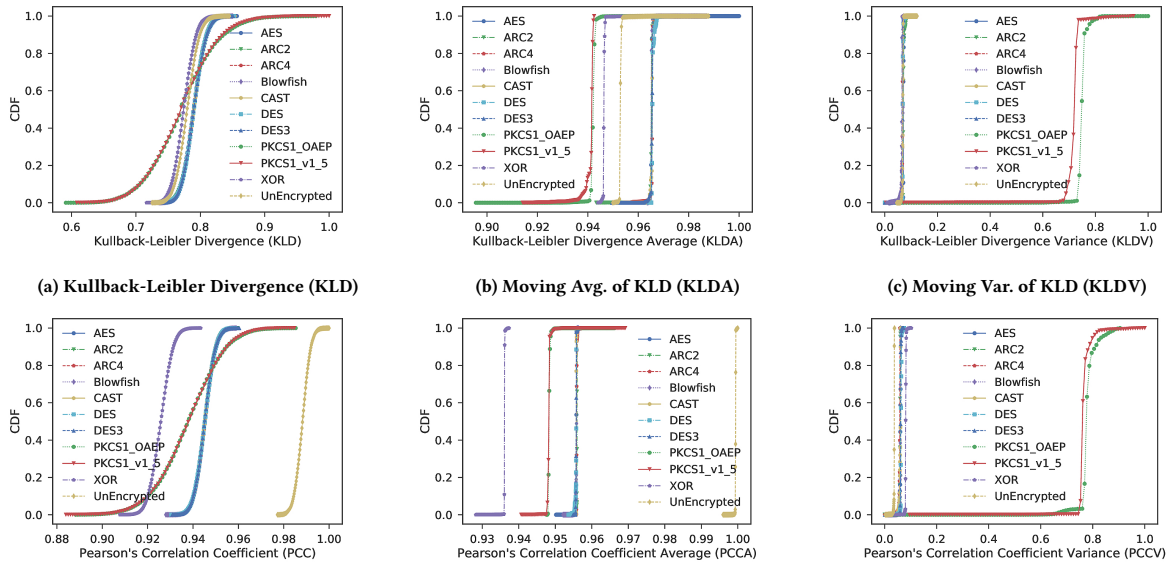
The features derived from KLD, i.e. KLDA and KLDV, corresponding to the n^{th} packet in the traffic flow, are defined as follows:

$$KLDA_n = \frac{1}{n}\sum_{i=0}^{i=n} KLD_i \quad (5) \qquad KLDV_n = \frac{1}{n}\sum_{i=0}^{i=n} (KLD_i - KLDA_i)^2 \quad (6)$$

2.4 Cipher and Key-Size Classification

In this subsection, we present two validation experiments, i.e., cipher classification (expt. 1) and key-size classification (expt. 2), to establish the efficacy of the aforementioned bit-sequence based features.

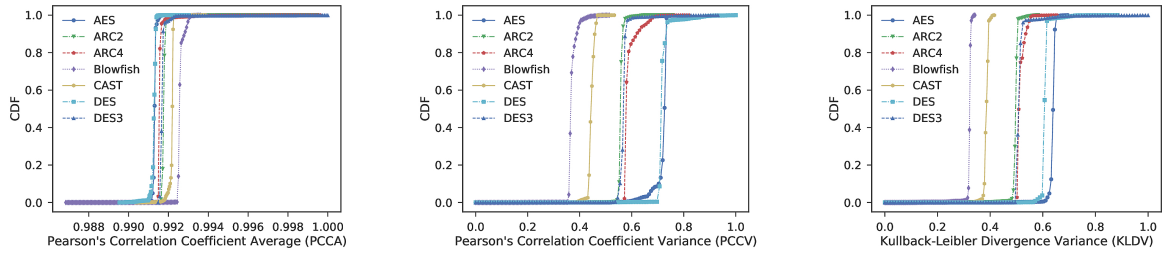
Ciphers: For expt. 1, we consider the same sources as in § 2.2, i.e., 7 symmetric-key ciphers (e.g., AES, ARC2, ARC4, Blowfish, CAST,



(a) Kullback-Leibler Divergence (KLD) (b) Moving Avg. of KLD (KLDA) (c) Moving Var. of KLD (KLDV)

(d) Pearson's Correlation Coeff. (PCC) (e) Moving Avg. of PCC (PCCA) (f) Moving Var. of PCC (PCCV)

Figure 2 Distribution (CDF) of bit-sequence based features: All 6 features – KL divergence (KLD), moving average of KL divergence (KLDA), moving variance of KL divergence (KLDV), Pearson's correlation coefficient (PCC), moving average of Pearson's correlation coefficient (PCCA), and moving variance of Pearson's correlation coefficient (PCCV) – are defined on the FFTs of the bit-sequences of packet payloads. These features capture the similarities (or dissimilarities) among bit-sequences of packet payloads generated using different ciphers (note that PKCS#1 v1.5 and PKCS#1 OAEP are represented as PKCS1_v1_5 and PKCS1_OAEP, respectively, in the plots).



(a) Moving Avg. of PCC (PCCA) (b) Moving Avg. of KLD (KLDA) (c) Moving Var. of KLD (KLDV)

Figure 3 Distribution (CDF) of bit-sequence based features PCCA, KLDA, and KLDV for 7 encryption algorithms, namely AES, ARC2, ARC4, Blowfish, CAST, DES, DES3. It can be observed that the packets generated for one encryption algorithm can be differentiated from others, on the basis of these 3 features.

DES, and DES3), 2 asymmetric ciphers (e.g., PKCS#1 OAEP and PKCS#1 v1.5), the XOR cipher, and unencrypted data acting as the 11^{th} source. We consider only the strongest cipher, i.e., AES, for expt. 2.

Key-sizes: In expt. 1, we use a fixed 128 bit key for symmetric ciphers, and a RSA public key extracted from a 1024 bit public/private key pair for asymmetric ciphers, as described in § 2.2. For expt. 2, we consider 3 different key-sizes of AES, i.e., 128 bits, 192 bits, and 256 bits. The keys are randomly generated using the Python `random.SystemRandom` class.

Setup: As in § 2.2, we used the Python package `PyCrypto` to encrypt each synthesized packet [3]. For both expts. 1 and 2, we generated 10,000 packets from each aforementioned source. In case of expt. 1, we repeated the experiments for 3 variations in terms of packet-size: (1) equal packet-size for all 11 sources, (2) equal packet-size for each source (different for different sources), and (3) random packet-size for each packet, irrespective of the packet source.

Results for cipher classification: We present the classification results for 8 different classifiers in Table 2. Multi-class (11 classes) classification is performed, all 6 features based on bit-sequence similarity are used, and 10-fold cross validation is employed to guard against over-fitting. We observe that the Random Forest Classifier (RFC) achieves best classification performance, with a worst-case accuracy of 89%. We also observe that the distribution of packet-sizes has an impact on classification, i.e., the packets can be more easily distinguished (96% in the best case) when packet-sizes are same across the packets from one source but different from the packets of other sources.

Results for key-size classification: Best performance is once again achieved by RFC, with an accuracy of 83% (with precision,

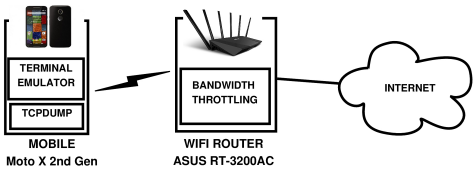


Figure 4 In-laboratory testbed setup for data collection: An ASUS RT-3200AC router connected to the Internet, acts as the gateway device. Bandwidth throttling is performed by setting router configuration parameters appropriately. A Moto X 2^{nd} generation Android smartphone running on Android 6.0 (Marshmallow) is connected to the router over WiFi. Mobile applications are executed on this phone, and data is collected using the *tcpdump* tool.

recall, and F1-score all at 0.83). We omit rest of the results (which are similar as for expt. 1) for brevity.

2.5 Bit-Sequence Features: Detailed Analysis

Having established that our proposed features are able to distinguish among data encrypted with different ciphers and different key-sizes, we perform a deeper analysis of those features. We plot the CDF for all 6 features for packets generated using each of the 11 sources in Fig. 2 (case (2), i.e., equal packet-size for each source is considered; the observations are similar in the other 2 cases). We observe that the distributions for PKCS#1 OAEP, PKCS#1 v1.5, XOR, and unencrypted can be easily differentiated from the rest of the cases (KLDV and PCCA showing the most prominent distinguishability). In order to analyze the other 7 encryption algorithms more closely, we normalize for those and plot the CDF of PCCA, PCCV, and KLDV in Fig. 3. We observe that even these 7 algorithms can be easily distinguished from each other with a classifier of higher arithmetic precision, on the basis of these 3 features.

So far, we have analyzed the effectiveness of our proposed set of features in classifying data encrypted with multiple ciphers/key-sizes. However, this ability is not merely limited to varying ciphers and key-sizes, but generalizes to any customizable parameter in a cryptosystem implementation that may give rise to a discernible difference in the randomness of the encrypted data. Other parameters (besides cipher and key-size) include the default OS library, the TLS library and its version being used, etc. In the following sections, we illustrate how these bit-sequence based features may be used to classify encrypted data generated from a large number of mobile apps, based on their ability to leverage variability in cryptosystem implementations.

3 EXPERIMENTAL SETUP AND DATASET

In this section, we explain our experimental setup for performing large-scale app traffic classification, and also describe the dataset collected using this setup.

3.1 Experimental Setup

We describe the experimental setup adopted by us, and the driving factors behind it, in this subsection.

Setup considerations: In order to generalize our results on mobile app traffic, we collected traffic traces under controlled operating conditions. Important control parameters included background traffic, channel conditions, and traffic from advertisements embedded in the traffic of the intended application. Even if all possible care is taken to avoid background traffic, it can still occur from services that the OS runs for optimizations and communications with the server. Also, variation in channel conditions may lead to differences in the way certain apps behave, e.g., video streaming apps may adapt to a different bit-rate when bandwidth varies. Similarly, advertisements running within the app may contaminate data collected from it. Keeping these factors in mind, we implemented an in-laboratory testbed to enable controlled collection of packet traces from a large number of mobile applications.

Testbed setup: The testbed setup is shown in Fig. 4. The end device connects to the ISP's network through a wireless router. The wireless router is instrumented such that we can control the bandwidth between the router and the end device. We used a Motorola Moto X 2^{nd} generation Android smartphone as the end device where we execute each mobile app separately. This device is rooted to allow trace collection using `tcpdump`. Android `Terminal Emulator` app provides the interface to execute `tcpdump`. The router is an ASUS RT-3200AC router with IEEE 802.11ac Wi-Fi standard. This router provides the *Adaptive QoS* feature, which allows manual configuration of the *Download Bandwidth* field to specify a controlled download rate.

Bandwidth scenarios: We collect data for each app under 4 bandwidth scenarios (by throttling at the router) – 0.5 Mbps, 1 Mbps, 4 Mbps, and 16 Mbps – to capture changes in app behaviour, if any, under varying bandwidth conditions.

Logs maintained during trace collection: We log the following information during data collection:

- ***Log 1:*** IPs for all client devices used in our experiments, and the corresponding app playing on each IP.
- ***Log 2:*** Start and end time for embedded advertisements, during execution of the app.

The aforementioned information enables us to discard noisy data from the accumulated traffic, in the manner described next.

Packet labeling: In order to associate each packet with a distinct label for our experiments, we first differentiate the video traffic flows based on 5-tuple information <*source IP, source port, destination IP, destination port, transport protocol*>, present in the header of each packet. Based on our knowledge of client IP addresses from ***Log 1***, we selectively preserve flows which correspond to download traffic, i.e., from the content server to the client app. We ensure that network control packets are eliminated from our dataset – such packets include ARP, DNS, IGMP, ICMP, NTP, DB-LSP-DISC (Dropbox Lan Sync Protocol), SYN, ACK (non-piggybacked), etc. – these packets do not carry any useful content, but appear in large numbers in the traces. Furthermore, we utilize the logged timing information in ***Log 2*** to discard packets which do not correspond to the actual app content, but to associated events (such as ads). Subsequently, we tag each packet with its source application, based on our knowledge of the client IP-app correspondence – cumulatively, this forms our dataset.

3.2 Android Application Dataset

In this subsection, we describe the dataset collected by us using the setup introduced in § 3.1.

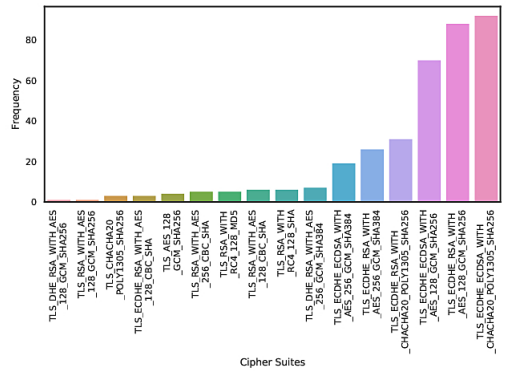


Figure 5 Distribution of cipher suites used by applications in our dataset.

Android app selection: We select 175 Android applications based on the *Top Free in Android Apps* selection in the Google Play Store[3]. The apps comprise a variety of categories, including but not limited to, video, audio, games, messaging, social networking, news, health, and static content-based apps.

Diversity in cipher suites: Razaghpanah et al. established in their analysis that different applications in the Android app store use different cipher suites, TLS libraries, OS defaults, etc. [41]. We perform a similar analysis of our dataset in terms of ciphers suites used and supported. We segregate out the `Server Hello` and `Client Hello` TLS handshake messages from our dataset, and parse them to find out cipher suites used by different applications. Fig. 5 shows the distribution of cipher suites used for encrypting traffic during the course of data collection. Fig. 6 on the other hand, shows the distribution of cipher suites which are supported by the application and may be used depending upon platform support, preference, or random choice. We observe from the figures that there is significant diversity in the cipher suites used for different mobile apps, which corroborates with the analysis in [41]. Furthermore, it is also important to note that cipher suites are not the only encryption parameters which affect randomness induced in the data; OS and TLS libraries, various configurable parameters on the client and server sides, etc. also impacts it. Our features are designed to leverage any such randomness occuring due to any combination of parameters in the cryptographic implementation of an application.

Choosing pairs of packets: We mentioned in § 2.3 that the ideal way of choosing pairs of packets in order to compute our proposed bit-sequence based features is to choose consecutive packets in a flow. We employ this idea (of consecutive packets) in the experiments conducted in this paper. The rationale is that in a real system, even if the client port is common (80 for HTTP and 443 for HTTPS) and the IPs are unrecognized for multiple simultaneously running applications, it would still be possible to segregate a flow based on its 5-tuple combination in most situations. There may be two exceptions, however: (1) when the first packet in a flow is encountered and it cannot be readily attributed to an application, and (2) when simultaneous flows end up with the same 5-tuple combination (e.g., when using a proxy). In both cases, we propose storing and using recent *reference* packets from all candidate applications, and using those for computation of the bit-sequence based features. The application for which the values correlate most closely (as returned by a trained classifier), is the likely target application.

In order to illustrate the frequency of case 1 (first packet of a flow) in our dataset, we plot the number of packets per flow for each application in Fig. 7. We observe that most (above 90%) of apps send atleast 100 packets in a flow, while around 50% send at least 1,000 packets. Around 15% of applications send upward of 10,000 packets in a single flow. This illustrates that in case of most applications the computation of bit-sequence based features does not incur any additional overhead, i.e., first packet instances are few and far between.

[3]This list changes frequently based on Google's app ranking system. Our selection was based on the status of this list on 15-April-2018.

4 EVALUATION

In this section, we evaluate the efficiency of our proposed bit-sequence based features in classifying traffic from a large number of applications.

4.1 Baseline (Packet-Level) Features

We introduce packet-level features in this subsection, such as those based on packet-size and packet inter-arrival time, which have been widely used in the literature for app-traffic classification [7, 27, 42, 46]. These features serve as baselines in our study. Along with the raw packet-size and inter-arrival time, we also consider the moving average and moving variance of these features. This is to account for sudden changes in the raw values, e.g., when the last packet in a burst is sent, it may be of an uncharacteristically lower size, or the first packet in a burst may have a significantly higher inter-arrival time. Moving average and variance values, which are more regular and smoothened, better capture the characteristic nature of app traffic.

Packet-Size Based Features: We use the following packet-size based features:

(1) *Packet-size (PS)*, which is directly the packet-size observed at the network (IP) layer.
(2) *Packet-size Moving Average (PSMA)*, which is the moving average of PS values for a traffic flow.
(3) *Packet-size Moving Variance (PSMV)*, which is the moving variance of PS values for a specific traffic flow.

PSMA and PSMV corresponding to the n^{th} packet in sequence of traffic flow, are formulated as follows:

$$PSMA_n = \frac{1}{n}\sum_{i=0}^{i=n} PS_i \quad (7) \qquad PSMV_n = \frac{1}{n}\sum_{i=0}^{i=n} (PS_i - PSMA_i)^2 \quad (8)$$

Packet Inter-Arrival Time Based Features: We use the following parameters based on temporal traffic behavior:

(1) *Inter-arrival time (IAT)*, which is merely the difference between arrival time of two consecutive packets belonging to the same flow.
(2) *Inter-packet timing average (IPTA)*, which is the moving average of IAT values for a traffic flow.
(3) *Inter-packet timing variance (IPTV)*, which is the moving variance of IAT values for a particular traffic flow.

The features identified above, i.e., IAT, IPTA, and IPTV, of the n^{th} packet in sequence within the flow, are formulated as follows:

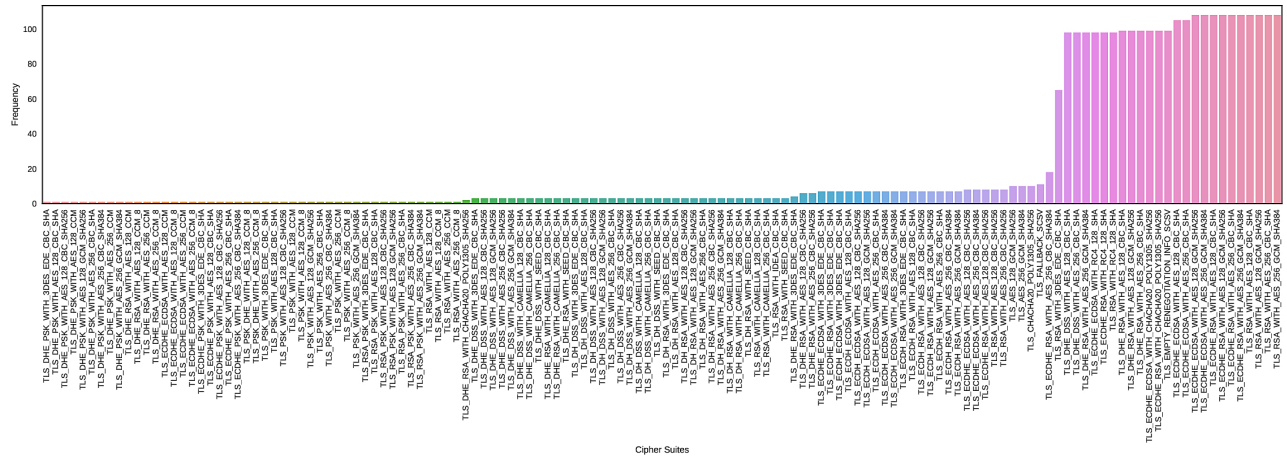


Figure 6 Distribution of cipher suites supported by applications in our dataset.

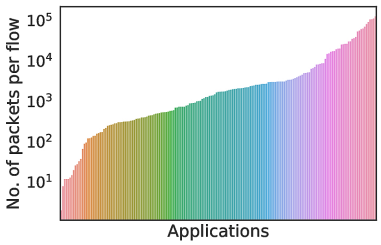


Figure 7 Distribution of number of packets per flow for applications

Table 3 Classification performance (precision ($\mathcal{P}$), recall ($\mathcal{R}$), f1-score ($\mathcal{F}$), accuracy ($\mathcal{A}$)) for classification of traffic from 175 mobile apps. Bit-sequence based features perform better than the baselines in terms of both F1-score and accuracy. The scores boost significantly when our proposed features are used in conjunction with baseline packet-level features.

Features / Classifier	**Random Forest**	**Decision Tree**
Metrics	$\mathcal{P}$, $\mathcal{R}$, $\mathcal{F}$, $\mathcal{A}$	$\mathcal{P}$, $\mathcal{R}$, $\mathcal{F}$, $\mathcal{A}$
Packet-Size Features (PSF)	0.32, 0.27, 0.27, **0.27**	0.35, 0.29, 0.29, **0.29**
Inter-Arrival Time Features (ITF)	0.33, 0.10, 0.12, **0.10**	0.37, 0.11, **0.13**, **0.11**
Packet-Level Features (PSF+ITF)	0.76, 0.62, 0.65, **0.62**	0.78, 0.60, **0.64**, **0.60**
Bit-Sequence Features (BSF)	0.68, 0.71, 0.69, **0.71**	0.66, 0.69, 0.67, **0.69**
Bit-Sequence + Packet-Level Features (PSF+ITF+BSF)	0.97, 0.96, 0.96, **0.96**	0.95, 0.95, 0.95, **0.95**

$$IAT_n = t_n - t_{n-1} \quad (9) \qquad IPTA_n = \frac{1}{n}\sum_{i=0}^{i=n} IAT_i \quad (10)$$

$$IPTV_n = \frac{1}{n}\sum_{i=0}^{i=n} (IAT_i - IPTA_i)^2 \quad (11)$$

where t_n and t_{n-1} are arrival time instances for the n^{th} and $(n-1)^{th}$ packets respectively.

4.2 Classification Methodology

We discuss the classification methodology employed for classification of packets from 175 mobile apps, followed by results.

Overcoming class imbalance: We train our classifiers on 10,000 packets from each app. In order to avoid bias due to class imbalance, we random-sample an equal number of packets (10,000) from the data collected for all those apps, where the total number of packets is higher than 10,000. In the few cases (5 apps) where lesser than 10,000 packets are available, we employ SMOTE [14] and oversample the datasets, so as to balance them.

Guarding against overfitting: In order to minimize the chances of over-fitting the data, we employ k-fold cross validation (we set $k = 10$ in our experiments).

4.3 Classification Results

Overall classification: The classification results for data collected from 175 apps are presented in Table 3. We present results for the following baselines: (1) Packet-size based features only (PS, PSMA, PSMV), (2) Inter-arrival time based features only (IAT, IPTA, IPTV), (3) Bit-sequence based features only (PCC, PCCA, PCCV, KLD, KLDA, KLDV), and (3) All packet-level features, i.e., a combination of packet-size based features and inter-arrival time based features (PS, PSMA, PCCV, IAT, IPTA, IPTV). Results for only the two best performing classifiers (Random Forest and Decision Tree) have been reported. We observe that using only packet-size based features results in an accuracy score of 0.29, while only inter-arrival time based features result in an ever lower accuracy of 0.11 (in the best cases, i.e., using the Decision Tree classifier). When combined, however, the packet-level features together result in a much higher accuracy of 0.60, which is still, however, not satisfactorily high.

Our proposed bit-sequence based features perform significantly better than either packet-size based or inter-arrival time based features, in terms of both F1-score and accuracy. In fact, they perform even better than all packet-level features combined (69% vs. 65% in terms of F1-score, and 71% vs. 62% in terms of accuracy). Considering that we perform multi-class classification with 175 classes, this is a significant result, and substantiates the effectiveness of our proposed features in differentiating apps based on differences in TLS implementations. The classification results are boosted heavily – in fact to 95% F1-score as well as accuracy, in the best case – when packet-level features are employed in conjunction with bit-sequence based features, which is our recommended strategy for mobile app traffic classification. A classifier performing with such

accuracy may be used by a network administrator in fingerprinting and monitoring heterogenous traffic from a vast number of apps.

Category-wise evaluation: We evaluate our proposed features for another important practical use-case. Instead of aggregate traffic, a network administrator may only be interested in identifying traffic from a specific category. For example, in an enterprise setting, she may try to enforce a policy where all video traffic is allocated lesser bandwidth to avoid overburdening critical applications. In this vein, we divide the entire dataset of apps into 5 categories, namely, video, audio, browsing, gaming, and messaging. The *browsing* category consists of apps used for email, download, sync, etc. apart from web browsing; the other 4 category names are self-explanatory. We then attempt to perform inter-category classification (i.e., classify among the 5 categories), as well as intra-category classification (i.e., classify among apps belonging to one particular category). The classification results are presented in Table 4.

We observe that we can achieve 90% accuracy (using only bit-pattern based features) in classifying heterogenous traffic into categories (row 1). Classification performance further improves when traffic from apps within a particular category are classified; minimum accuracy (using RFC) is 93%, which once again is a significant results considering the scale of classification.

Classification running time: Since the classifier needs to run as part of a live system, it is also important to evaluate the running time for per-packet classification. We executed a separate batch of experiments, where we trained the classifiers with 90% of the data, and tested with the remaining 10%. This enabled fine-grained time-keeping, which we achieved using the Python `timeit` module. These experiments were executed in a Ubuntu 16.04 workstation, with 64 GB RAM, and a 3.0 GHz Intel Core i5 processor. We observe that while RFC consumed 14.16 microseconds per packet, the Decision Tree classifier consumed only 0.84 microsecond. We note that these time quanta are well within the bounds of the RTT of packets arriving from a distant content server, which implies that our classification strategy can be employed in a real-life setting, such as in a smart-QoS enabled modern home router.

5 DISCUSSION

In this section, we discuss some of the major learnings from and limitations of this work, as well as the potential impact it may have in inducing further research in the area.

Real-time implementation: We show in § 4.3 that classification at per-packet granularity is compliant with the timing requirements of a live networked system. However, we do not comment on the time consumed in the training process, when either training frequency is high, or training data volume is extremely large. For example, network administrators intent on classifying and blocking malicious apps may need to avail frequent rounds of training, since new apps crop up everyday on the app store. Optimizations such as implementing FFT in hardware could be availed to speed up computations in such scenarios [6].

Attacks around bit-sequence based features: While our use-cases in this paper revolve around the opportunities bit-sequence based features could provide network administrators, the flip-side of the coin is that the same opportunities may be exploited by attackers. For example, these features could be leveraged to infer user actions from mobile phone traffic, which could lead to severe breach of privacy for users [18, 37, 49]. Applications which authenticate users based on their app activity history [21, 30], may also be fooled if app activities could be gleaned out using our proposed features.

Protection against obfuscation: Some works have pointed out the privacy risks involved when statistical analysis of packet-level features is used to differentiate traffic [32, 51]. These studies, including a recent work [13] propose obfuscating such features so that classifiers may not be able to extract useful information out of application traffic. The obfuscation is implemented in a proxy, which fragments incoming packets and regulates arrival rate to make one class of traffic look like another. Our classification mechanism based on randomness of security implementations, is however, immune to such packet-level obfuscation.

Tool for side-channel attacks: It follows from the previous discussion, therefore, that along with packet-level and traffic meta-data (e.g., direction of traffic) based features [17, 45, 48], bit-sequence based features can also pose threats via side-channel attacks. It is important therefore, to further the research agenda in this direction, and devise methods to either exploit or alleviate such attacks, as necessary for the application.

Impact of app updates: App updates may include updates to their encryption implementations as well. While such updates can affect classification performance of our trained model, as long as there exist differences in such implementations, it is possible to retrain the classifier to recover performance. We argue that such differences will continue to exist due to the non-standard nature of app stores, and deficiencies in computational capabilities of low-end devices.

6 RELATED WORK

Classifying traffic from mobile applications has spurred active research recently due to its practical necessity, as well as, the challenges arising from lack of features suitable for traffic classification. First, we look at classification strategies which have addressed generic web traffic. Next, we present studies specific to mobile traffic, and discuss their advantages and limitations.

6.1 Generic Web Traffic Classification

A large body of work has addressed traffic classification on the Internet over a number of years. Continuously changing landscapes in the way data is delivered to the end-users (e.g., content delivery networks (CDNs), cloud), have prompted researchers to revisit old techniques, and invent newer approaches more commensurate with the trends prevalent during the time of their studies.

Deep Packet Inspection (DPI) based Techniques: Several approaches have been suggested for web traffic classification using packet-inspection based methods. Finsterbusch et al. compiled a comprehensive survey of these techniques – however, they note, like many others, that DPI based approaches either perform abysmally, or fail entirely when traffic is encrypted [26].

Statistical Information based Techniques: Most studies which do not rely on packet-inspection, suggest exploiting side-channel information, and statistical properties of traffic, which remain unaffected by encryption. An MIT tech review article highlighted the importance of statistical tricks in extracting user footprints from encrypted communications [5]. Some statistical approaches rely on

Table 4 Classification results (precision ($\mathcal{P}$), recall ($\mathcal{R}$), f1-score ($\mathcal{F}$), accuracy ($\mathcal{A}$)) using only bit-sequence based features for 2 classification tasks: (1) classification of aggregate traffic into app categories (shown in the first row), and (2) classification of category traffic into constituent applications. Best classification performance is achieved using the Random Forest Classifier (highlighted in green).

Clf. task / Classifier	Random Forest	Decision Tree	K-Nearest Neighbors	Naive Bayes	M'layer Perceptron (Deep)	Logistic Regression	Support Vector
Metrics	$\mathcal{P}, \mathcal{R}, \mathcal{F}, \mathcal{A}$	$\mathcal{P}, \mathcal{R}, \mathcal{F}, \mathcal{A}$	$\mathcal{P}, \mathcal{R}, \mathcal{F}, \mathcal{A}$	$\mathcal{P}, \mathcal{R}, \mathcal{F}, \mathcal{A}$	$\mathcal{P}, \mathcal{R}, \mathcal{F}, \mathcal{A}$	$\mathcal{P}, \mathcal{R}, \mathcal{F}, \mathcal{A}$	$\mathcal{P}, \mathcal{R}, \mathcal{F}, \mathcal{A}$
App categories (5)	0.90, 0.90, 0.90, **0.90**	0.88, 0.88, 0.88, **0.88**	0.77, 0.77, 0.77, **0.77**	0.47, 0.46, 0.41, **0.46**	0.42, 0.47, 0.42, **0.47**	0.38, 0.42, 0.36, **0.42**	0.37, 0.40, 0.37, **0.40**
Video	0.95, 0.95, 0.95, **0.95**	0.95, 0.95, 0.95, **0.95**	0.72, 0.72, 0.72, **0.72**	0.33, 0.30, 0.27, **0.30**	0.27, 0.29, 0.27, **0.29**	0.22, 0.19, 0.15, **0.19**	0.37, 0.40, 0.37, **0.40**
Audio	0.99, 0.99, 0.99, **0.99**	0.98, 0.98, 0.98, **0.98**	0.91, 0.90, 0.90, **0.90**	0.39, 0.35, 0.32, **0.35**	0.46, 0.43, 0.42, **0.43**	0.41, 0.38, 0.36, **0.38**	0.22, 0.20, 0.12, **0.20**
Browsing	0.94, 0.93, 0.93, **0.93**	0.92, 0.92, 0.92, **0.92**	0.68, 0.69, 0.68, **0.69**	0.20, 0.19, 0.16, **0.19**	0.13, 0.19, 0.13, **0.19**	0.16, 0.16, 0.13, **0.16**	0.22, 0.21, 0.22, **0.21**
Gaming	0.96, 0.96, 0.96, **0.96**	0.94, 0.94, 0.94, **0.94**	0.61, 0.62, 0.60, **0.62**	0.14, 0.15, 0.12, **0.15**	0.11, 0.12, 0.10, **0.12**	0.13, 0.14, 0.12, **0.14**	0.11, 0.10, 0.11, **0.12**
Messaging	0.99, 0.99, 0.99, **0.99**	0.98, 0.98, 0.98, **0.98**	0.91, 0.91, 0.91, **0.91**	0.65, 0.52, 0.52, **0.52**	0.39, 0.41, 0.35, **0.41**	0.48, 0.42, 0.38, **0.42**	0.46, 0.41, 0.38, **0.41**

carefully constructed rules for classification, while others employ machine learning – we report these cases separately.

Rule based Approaches: Sun et al. presented a method for statistical identification of browsed web applications, by studying the various unique web objects (e.g., images, stylesheets, etc.) fetched during the launch of a webpage [47]. This approach is clearly outdated, since most websites serve dynamic content nowadays, where fetched objects would continually change, sometimes even within seconds. Gong et al. proposed using side-channel attacks to determine round-trip times (RTTs), and then clustering traffic with similar RTTs [29]. A similar approach was adopted by Chen et al., where side-channel leaks in terms of packet-size, arrival time, and stateful interactions between servers and clients of web applications, were exploited [15]. Piskac et al. suggested protocol detection from web traffic, again using packet-size and arrival time values [40]. Detection of Domain Generation Algorithm (DGA) based botnets was proposed using DNS traffic analysis in a recent work by Wang et al. [50]. Another recent work studied privacy vulnerabilities in a IoT based smart home system, made possible by traffic analysis [9].

Machine Learning based Approaches: Traffic classification using supervised learning has been a popular approach – the major task in such an approach is to identify discriminative features from encrypted network traffic. Some of the earlier works can be found in a survey by Nguyen et al. in [39], and references therein. Jesudasan et al. identified generic features (based on packet-size and arrival time) across different versions of Skype that could be used to classify Skype flows [33]. Bujlow et al. proposed a learning model with features based on individual and aggregate payload sizes, along with IP and port no. based rules [11]. Recently, Shi et al. have also shown that encrypted, and even tunneled (using a Virtual Private Network (VPN)) video sources, can be identified by analyzing simple features derived from packet-sizes and temporal patterns of packet arrival [10, 43]. As an application of network traffic analysis, a recent work by Das et al. proposed privacy-aware localization using a supervised learning approach [22].

While some statistical observations for generic web traffic may still hold true in case of mobile traffic, there are differences due to buffer and bandwidth constraints on a mobile device, and how service providers tend to optimize traffic for mobile accordingly. We leverage upon the common statistical properties by identifying those (with careful analysis of mobile app traffic), and address the differences by defining novel features that can aid in classification.

6.2 Mobile Traffic Classification

Studies on mobile traffic classification can be broadly divided based on whether those targeted unencrypted or encrypted traffic.

Unencrypted Traffic: Approaches that assume unencrypted traffic rely on inspection of fields in the HTTP header. Hur et al. exploited the user agent field, and other common strings found in the HTTP headers of packets generated by the same application [31]. Dai et al. showed that by executing different apps in an emulator and gathering corresponding network traces, it is possible to build signatures for identifying specific traffic sources [20]. Yao et al. reduced the overhead of building signatures where they used context of signatures to improve identification accuracy and scalability [54]. Automatic generation of classifiers for traffic identification has been presented in [16, 53]. AppPrint also uses extensive traffic observations to build signatures based on header fields [38]. Since these works rely on information that must be read from the packets, they are ill-suited for classifying encrypted traffic.

Encrypted Traffic: Stöber et al. proposed classification of background traffic generated from a smartphone, by studying patterns in packet level features (e.g., packet-size and packet arrival instances), and burst level features (e.g., mean, median, etc. of packet level features) [46]. Alan et al. limited their study to using only launch time traffic for building supervised classifiers for Android app identification [7]. Mobile messaging apps were studied by Fu et al., and a machine learning based solution was provided using packet-level features [27]. Another work defined an end-to-end system to enable fine-grained bandwidth provisioning for video app traffic in absence of background traffic [42].

7 CONCLUSION

Mobile traffic classification is an important task for network administrators in order to achieve policy enforcement and service provisioning. However, due to lack of unique signatures and especially due to encryption, mobile app traffic classification is rendered difficult. While statistical analysis based on packet-size and arrival time of packets has shown a lot of promise, existing techniques do not scale well to traffic from a large number of applications. In this paper, we worked on the notion of exploiting encryption itself (more specifically, TLS implementations) to classify mobile app traffic at a large scale. We proposed a set of novel bit-sequence based features, which can classify data encrypted with different ciphers and key-sizes, with 89% and 83% accuracy (best case), respectively. We show that using these features in conjunction with existing packet-level based features, it is possible to classify traffic from 175 Android applications with a best case accuracy of 95% (as opposed to a baseline accuracy of 62%). The features also perform well in achieving inter-category (90% accuracy) and intra-category traffic classification (min. accuracy of 93%), using only bit-sequence based features.

REFERENCES

[1] [n. d.]. Cisco Visual Networking Index White Paper. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

[2] [n. d.]. Github Library: r4nd0m. https://github.com/StuartGordonReid/r4nd0m

[3] [n. d.]. Package Crypto: Python Cryptography Toolkit. https://www.dlitz.net/software/pycrypto/api/current/

[4] [n. d.]. Random Bit Generation: Guide to the Statistical Tests. https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software/Guide-to-the-Statistical-Tests

[5] [n. d.]. Statistical Tricks Extract Sensitive Data from Encrypted Communications.

[6] Berkin Akin, Franz Franchetti, and James C Hoe. 2014. Understanding the design space of dram-optimized hardware FFT accelerators. In *Application-specific Systems, Architectures and Processors (ASAP), 2014 IEEE 25th International Conference on*. IEEE, 248–255.

[7] Hasan Faik Alan and Jasleen Kaur. 2016. Can Android Applications Be Identified Using Only TCP/IP Headers of Their Launch Time Traffic?. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 61–66.

[8] Mohammed M Alani. 2010. Testing randomness in ciphertext of block-ciphers using DieHard tests. *Int. J. Comput. Sci. Netw. Secur* 10, 4 (2010), 53–57.

[9] Noah Apthorpe, Dillon Reisman, and Nick Feamster. 2017. A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic. *arXiv preprint arXiv:1705.06805* (2017).

[10] Subir Biswas and Yan Shi. 2016. Protocol independent identification of encrypted video traffic sources using traffic analysis. In *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 1–6.

[11] Tomasz Bujlow, Tahir Riaz, and Jens Myrup Pedersen. 2012. A method for classification of network traffic based on C5. 0 Machine Learning Algorithm. In *Computing, Networking and Communications (ICNC), 2012 International Conference on*. IEEE, 237–241.

[12] Arthur Callado, Carlos Kamienski, Géza Szabó, Balázs Péter Gerö, Judith Kelner, Stênio Fernandes, and Djamel Sadok. 2009. A survey on internet traffic identification. *Communications Surveys & Tutorials, IEEE* 11, 3 (2009), 37–52.

[13] Louma Chaddad, Ali Chehab, Imad H Elhajj, and Ayman Kayssi. 2018. App traffic mutation: Toward defending against mobile statistical traffic analysis. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE.

[14] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[15] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. 2010. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 191–206.

[16] Yeongrak Choi, Jae Yoon Chung, Byungchul Park, and James Won-Ki Hong. 2012. Automated classifier generation for application-level mobile traffic identification. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*. 1075–1081.

[17] Mauro Conti, Qian Qian Li, Alberto Maragno, and Riccardo Spolaor. 2018. The Dark Side (-Channel) of Mobile Devices: A Survey on Network Traffic Analysis. *IEEE Communications Surveys & Tutorials* (2018).

[18] Mauro Conti, Luigi V Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. 2015. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*. ACM, 297–304.

[19] RR Coveyou and Robert D MacPherson. 1967. Fourier analysis of uniform random number generators. *Journal of the ACM (JACM)* 14, 1 (1967), 100–119.

[20] Shuaifu Dai, Alok Tongaonkar, Xiaoyin Wang, Antonio Nucci, and Dong Song. 2013. Networkprofiler: Towards automatic fingerprinting of android apps. In *INFOCOM, 2013 Proceedings IEEE*. IEEE, 809–817.

[21] Sourav Kumar Dandapat, Swadhin Pradhan, Bivas Mitra, Romit Roy Choudhury, and Niloy Ganguly. 2015. Activpass: your daily activity is your password. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2325–2334.

[22] Aveek K Das, Parth H Pathak, Chen-Nee Chuah, and Prasant Mohapatra. 2017. Privacy-aware contextual localization using network traffic analysis. *Computer Networks* 118 (2017), 24–36.

[23] Ali Doganaksoy, Baris Ege, Onur Koçak, and Fatih Sulak. 2010. Cryptographic Randomness Testing of Block Ciphers and Hash Functions. *IACR Cryptology ePrint Archive* 2010 (2010), 564.

[24] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. 2010. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 281–287.

[25] Horst Feistel. 1973. Cryptography and Computer Privacy. *Scientific American* 228, 5 (1973), 15–23.

[26] Michael Finsterbusch, Chris Richter, Eduardo Rocha, Jean-Alexander Muller, and Klaus Hanssgen. 2014. A survey of payload-based traffic classification approaches. *Communications Surveys & Tutorials, IEEE* 16, 2 (2014), 1135–1156.

[27] Yanjie Fu, Hui Xiong, Xinjiang Lu, Jin Yang, and Can Chen. 2016. Service usage classification with encrypted internet traffic in mobile messaging apps. *IEEE Transactions on Mobile Computing* 15, 11 (2016), 2851–2864.

[28] Aaron Gember, Ashok Anand, and Aditya Akella. 2011. A comparative study of handheld and non-handheld traffic in campus Wi-Fi networks. In *Passive and Active Measurement*. Springer, 173–183.

[29] Xun Gong, Negar Kiyavash, and Nikita Borisov. 2010. Fingerprinting websites using remote traffic analysis. In *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 684–686.

[30] Alina Hang, Alexander De Luca, and Heinrich Hussmann. 2015. I know what you did last week! do you?: Dynamic security questions for fallback authentication on smartphones. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1383–1392.

[31] Min Hur and Myung-Sup Kim. 2012. Towards smart phone traffic classification. In *Network Operations and Management Symposium (APNOMS), 2012 14th Asia-Pacific*. IEEE, 1–4.

[32] Alfonso Iacovazzi and Andrea Baiocchi. 2014. Internet traffic privacy enhancement with masking: Optimization and tradeoffs. *IEEE Transactions on Parallel and Distributed Systems* 25, 2 (2014), 353–362.

[33] Rozanna Nadeera Jesudasan, Philip Branch, and Jason But. 2010. Generic attributes for Skype identification using machine learning. *Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. A* 100820 (2010), 20.

[34] John B. Kam and George I. Davida. 1979. Structured Design of Substitution-Permutation Encryption Networks. *IEEE Trans. Comput.* 10 (1979), 747–753.

[35] Vasilios Katos. 2005. A randomness test for block ciphers. *Applied mathematics and computation* 162, 1 (2005), 29–35.

[36] Donald E Knuth. 1981. *The Art of Programming, vol. 2, Semi-Numerical Algorithms*. Addison Wesley, Reading, MA.

[37] Conti Mauro, Luigi Vincenzo Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. 2016. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions on Information Forensics and Security* 11, 1 (2016), 114–125.

[38] Stanislav Miskovic, Gene Moo Lee, Yong Liao, and Mario Baldi. 2015. AppPrint: Automatic Fingerprinting of Mobile Applications in Network Traffic. In *Proceedings of Passive and Active Measurement Conference*. 57–69.

[39] Thuy TT Nguyen and Grenville Armitage. 2008. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys & Tutorials, IEEE* 10, 4 (2008), 56–76.

[40] Pavel Piskac and Jiri Novotny. 2011. Using of time characteristics in data flow for traffic classification. *Managing the Dynamics of Networks and Services* (2011), 173–176.

[41] Abbas Razaghpanah, Arian Akhavan Niaki, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Johanna Amann, and Phillipa Gill. 2017. Studying TLS usage in Android apps. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. ACM, 350–362.

[42] Satadal Sengupta, Vinay Kumar Yadav, Yash Saraf, Harshit Gupta, Niloy Ganguly, Sandip Chakraborty, and Pradipta De. 2017. MoViDiff: Enabling service differentiation for mobile video apps. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*. IEEE, 537–543.

[43] Yan Shi and Subir Biswas. 2015. Characterization of Traffic Analysis based video stream source identification. In *Advanced Networks and Telecommuncations Systems (ANTS), 2015 IEEE International Conference on*. IEEE, 1–6.

[44] Juan Soto and Lawrence Bassham. 2000. *Randomness testing of the advanced encryption standard finalist candidates*. Technical Report. BOOZ-ALLEN AND HAMILTON INC MCLEAN VA.

[45] Raphael Spreitzer, Veelasha Moonsamy, Thomas Korak, and Stefan Mangard. 2018. Systematic classification of side-channel attacks: a case study for mobile devices. (2018).

[46] Tim Stöber, Mario Frank, Jens Schmitt, and Ivan Martinovic. 2013. Who do you sync you are?: smartphone fingerprinting via application behaviour. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*. ACM, 7–12.

[47] Qixiang Sun, Daniel R Simon, Yi-Min Wang, Wilf Russell, Venkata N Padmanabhan, and Lili Qiu. 2002. Statistical identification of encrypted web browsing traffic. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE, 19–30.

[48] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2018. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security* 13, 1 (2018), 63–78.

[49] Qinglong Wang, Amir Yahyavi, Bettina Kemme, and Wenbo He. 2015. I know what you did on your smartphone: Inferring app usage over encrypted data traffic. In *Communications and Network Security (CNS), 2015 IEEE Conference on*. IEEE, 433–441.

[50] Tzy-Shiah Wang, Hui-Tang Lin, Wei-Tsung Cheng, and Chang-Yu Chen. 2017. DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis. *Computers & Security* 64 (2017), 1–15.

[51] Charles V Wright, Scott E Coull, and Fabian Monrose. 2009. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis.. In *NDSS*, Vol. 9. Citeseer.

[52] Yue Wu, Joseph P Noonan, and Sos Agaian. 2011. NPCR and UACI randomness tests for image encryption. *Cyber journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT)* 1, 2 (2011), 31–38.

[53] Qiang Xu, Yong Liao, Stanislav Miskovic, Mario Baldi, Z. Morley Mao, Antonio Nucci, and Thomas Andrews. 2015. Automatic Generation of Mobile App Signatures from Traffic Observations. In *Proceedings of IEEE INFOCOM 2015 (INFOCOM '15)*.

[54] 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 '15)*. 439–451.