An Entropy Based Encrypted Traffic Classifier

Mohammad Saiful Islam Mamun(✉), Ali A. Ghorbani, and Natalia Stakhanova

Information Security Centre of Excellence (ISCX), University of New Brunswick, Fredericton, NB, Canada
msi.mamun@unb.ca, ghorbani@unb.ca, natalia.stakhanova@unb.ca

Abstract. This paper proposes an approach of encrypted network traffic classification based on entropy calculation and machine learning technique. Apart from using ordinary Shannon's entropy, we examine entropy after encoding and a weighted average of Shannon binary entropy called BiEntropy. The objective of this paper is to identify any application flows as part of encrypted traffic. To achieve this we (i) calculate entropy-based features from the packet payload: encoded payload or binary payload, *n-length* word of the payload, (ii) employ a Genetic-search feature selection algorithm on the extracted features where fitness function is calculated from True Positive Rate, False Positive Rate and number of selected features, and (iii) propose a data driven supervised machine learning model from Support Vector Machine (SVM) for automatic identification of encrypted traffic. To the best of our knowledge, this is the first attempt to tackle the problem of classifying encrypted traffic using extensive entropy-based features and machine learning techniques.

Keywords: Traffic classification · Entropy · Encoding

1 Introduction

Today's Internet is evolving in scope and complexity. Internet applications adapt rapidly to changing situations. Emerging proprietary protocols (e.g. Skype) serve as obfuscation to bypass network filters, firewalls and NAT restrictions. All these developments significantly obstruct efforts in network analysis and monitoring. Understanding network traffic content (e.g., intrusion detection purposes) becomes increasingly difficult task. Mostly because it requires requires an understanding of network traffic nature or characteristics that can associate traffic flows with the application types. The research studies show that existing traffic classification methods are plagued with problems [1]. Among them the lack of accuracy in classification and unavailability of datasets are prominent.

Techniques based on packet payloads (e.g., [17,20]), usually called Deep Packet Inspection (DPI), provide more accurate result since they classify the traffic based on comparing information included in the packet payload against existing signatures. But these techniques cannot be used in many deployment environments due to privacy concerns [29]. (e.g., imposed legal restrictions of

© Springer International Publishing Switzerland 2016
S. Qing et al. (Eds.): ICICS 2015, LNCS 9543, pp. 282–294, 2016.
DOI: 10.1007/978-3-319-29814-6_23

the organization) or unable to cope with completely encrypted network packets e.g., application is unknown or newly launched (no signatures exists).

Alternatively, techniques based on packet headers [2,3,10,13,14], usually called behavioral statistics focus on general properties of the packet headers/flows (such as packet size, port numbers, inter arrival times, average number of packets in a flow etc.) and aim to match a behavioral pattern to certain applications. However, these models' accuracy were evaluated on the test dataset that might fail when being applied to different point of metric and network environment e.g., network latency jitter, packet loss or fragmentation. Moreover, *time* based features employed by many of these techniques are less likely to remain the same across different networks.

A packet consisting of a sequence of random numbers or a compressed data[1] also yields high entropy level [33]. Therefore, high entropy of a packet alone does not always imply encryption. For instance, a compressed file such as, .gz or .zip shows a high entropy level. In the latter case, the data is actually highly structured and not random [22]. Therefore, observing ordinary entropy alone would not necessarily provide enough information to let the classifier distinguish between encrypted and compressed unencrypted traffic. In this context, we take a look at an alternative set of features.

Apart from encryption, encoding plays a significant role in achieving high-entropy. For example, if we just encode a random bytes into HEX, the resulting string will be twice (each byte encoded to HEX consists of 2 characters) as long as the original random bytes. It sharply increases the randomness and hence entropy. Besides that, many protocols use ASCII encoded plain text for unencrypted payload. If it is converted to HEX values, it looks nearly random. Therefore, HEX-encoded entropy of these payloads is very close to encrypted payload.

Our Contribution: Our DPI-based technique use machine learning (ML) based classifier for filtering out unencrypted traffic. To bypass compressed data, we focus on a binary entropy analysis tool, called Bintropy. Bintropy was introduced as an analysis tool which is a prototype analysis tool to estimate the likelihood of the compressed content [23]. Although the target was to explore trends associated with malware executable files in either compression or encryption state, we discover that features extracting from Bintropy with other reliable features significantly boost the overall performance of the classifier. Note that due to high-entropy output for encrypted traffic, a variant of Bintropy called TBiEn[2] has been used in the experiment.

In order to placate encoded plain text problem, we analyze the influence of the size of the alphabet (n-length) in encoded message on the entropy calculation. At this point, we found that n-length HEX-encoded entropy can distinguish encrypted payloads more correctly where ordinary HEX encoding fails. Since most of the payloads are received in HEX values, we then stress more on a

[1] Since the repeated patterns in the uncompressed data are removed the redundancy also disappears making data look random and less predictable.

[2] A logarithmic weighting that gives higher weight to the higher derivatives.

complex possibility that is to check for any other standard encoding scheme e.g., UTF-8.

To decrease training time and improve accuracy and performance of the classifier, we employ feature selection algorithm and choose the best and promising features extracted from different entropy based algorithms. We combine Genetic algorithm with Support Vector Machine (SVM) following the model presented in [34] where authors develop a detection system with optimal features for diagnosing diseases. By using the same model, our aim is to discover traffic patterns for the automatic identification of encrypted traffic from any anonymous network traffic.

Decision trees, in comparison to other classification techniques, produce a set of rules and using C5.0 algorithm can easily be incorporated into real time techniques like traffic classification. As an alternative approach to SVM-based model, we build a decision tree model, namely C5.0 [26] with optimal features and check the feature usage by C5imp function.

Finally, results from Genetic-SVM and C5.0, were cross-checked by employing an unsupervised machine learning algorithm such as K-Means [37]. Cluster analysis is a promising tool to identify traffic classes. We evaluate K-Means as an external validation measure for classifier models using identical features on the same dataset. In [36], authors applied DBSCAN and K-Means clustering algorithms to group Internet traffic using transport layer characteristics. The idea was to use *unsupervised* learning technique to group *unlabelled* training data based on similarity. This approach possesses some practical benefits over learning approaches with labelled training data [36].

The rest of this paper is organized as follows. Related work is discussed in Sect. 2. Section 3 briefly describes the dataset collection and pre-processing for extracting features. While Sect. 4 details employed machine learning algorithms, the experimental results, evaluation and validation methods, conclusions are drawn in Sect. 5.

2 Related Work

In this section, we discuss relevant work based on payloads in traffic classification. Then we focus on some recent work based on entropy in different field of the traffic classification.

DPI or payload based techniques [11,12,16,17] were mainly used for unencrypted traffic because of their error-prone nature to completely encrypted traffic. However, a very recent work on encrypted-traffic based DPI technique in [20] where authors present a classifier based on first-order homogeneous Markov chain to identify encrypted traffic for applications such as Dropbox, Twitter etc. However, if the application protocol do not follow the RFC specifications strictly, use SSL for tunneling only (e.g., Skype, Tor), or behave differently from SSL stacks, this technique fails abruptly. Besides that, the employed periodic update fingerprints need as the application nature changes nature over time.

In [30], an *statistical modelling* based approach has been proposed where the authors define a vector from the number of different *word-values* available in the

payload and match the similarity of the vectors with payloads from consecutive packets. After classifying the similarity between network flows they conclude with tracing five applications with 80 % accuracy. A pattern-matching P2P traffic classifier has been proposed in [31] that extracts payload signatures by using temporal correlation of flows and observing network traffic on different links. In [20], authors propose a payload-based method depending on first-order homogeneous Markov chains fingerprints conveyed in TLS sessions. Since application signatures change over time, their fingerprint models need to be updated periodically.

Entropy-based approaches are often used to detect malicious traffic [23–25]. A very few studies have been done for specific encrypted application e.g., Skype [19]. However, we found no research solely devoted to classifying encrypted and unencrypted traffic using entropy. Here we discuss some of the previous work using entropy in the literature.

In Olivian and Goubault-Larrecq [24], proposed an N-truncated entropy range for selected encrypted applications. If a payload's entropy does not satisfy the range, the traffic is deemed as malicious. The study considers consider the payload of the first packet of a connection and compares it with the N-truncated entropy of uniformly distributed payload.

Dorfinger et al. in [19] propose a real-time classifier based on the entropy of the first packet's payload of a flow and compare the result with the entropy of random payload. We found that in some encrypted applications or protocol first packet in a flow is partially encrypted (e.g. Client Hello Message of TLS) or smaller in size that may cause false positive result or unreliable evaluation [20]. Besides that it's an application (skype) specific solution.

In [21], Sun et al. investigated the encrypted web pages based on entropy in a large sample of Web pages using individual object sizes in a page to identify encrypted pages. In [15], authors studied traffic classification from several P2P and non-P2P applications where entropy was used to measure the heterogeneity of network level to characterize traffic samples. Zhang et al. in [18] proposed two high-entropy classifiers to detect bot. In order to avoid falsification, this scheme adds extra tools or mechanisms e.g., BotHunter and flow based cumulative entropy.

In [33], authors use BiEntropy to explore noticeable differences between the encrypted and unencrypted binary files of some real and synthetic spreadsheets (e.g., MS Excel).

Note that almost all of the aforementioned works are based on the measurement of conventional Shannon entropy over the whole packet's payload and for any specific protocol or application. We on the other hand focus on encrypted traffic in general. In compare to previous scheme, we extend our approach in mainly two ways: using different types of entropy measurements and employing it not for a specific protocol, but for any kind of encrypted traffic.

3 Dataset and Preprocessing

When starting a machine learning project the most important piece and hardest to obtain is to get a perfect data set that would have to be heterogeneous enough to emulate real traffic to some extent. To provide a comprehensive evaluation of encrypted traffic, we gathered a large collection of traffic samples representing both encrypted and unencrypted traces.

Capturing background network data in a real life environment is challenging due to two main reasons: privacy concerns of the gathered data and noise such as unwanted packets. Typically, traffic is either collected from a trusted sources or captured in a controlled environment. To address this challenge, our experimental dataset has been generated from combining non-overlapping portion of the several data sources where ground truth was known. Our accumulated dataset combines traffic traces mainly from ISCX 2015 dataset [7], skype traces from [8] and some traces from wireshark [9] samples.

- ISCX 2015 dataset [7] has been created in a controlled testbed environment simulating real applications from a diverse set of application traces and background traffic. To facilitate the labeling process or removing the need of sanitization, when capturing the traffic all unnecessary services and applications were closed. The traffic was captured with wireshark [9] and tcpdump [32], generating a total amount of 5 GB of data (encrypted/unencrypted). Protocols such as HTTPS, HTTP, SMTP, SSH, FTP were chosen to be simulated. However, following protocols were generated readily as a consequence of employing aforementioned protocols: DNS and NetBios. Specific applications were labelled accordingly, for example, Facebook, Twitter, Dropbox traffic as *secure web* or *https* connection. The prepared labelled dataset including full packet payloads, along with the relevant entropy results are publicly available to researchers through our website [7].
- Tstat dataset in [8], a research project testbed data containing only Skype traffic. Traces have been collected and organized with the support of the RECIPE (Robust and Efficient traffic Classification in IP nEtworks) projects.
- Sample captures in Wireshark wiki [9] are some example capture files collected from several application/protocol traces.

Since the dataset includes heterogeneous traffic flows (mixed traffic), experiments on datasets would help assess the accuracy of classification. However, even if the dataset containing *payloads* were collected from homogeneous traffic flows, they must be network independent in compared to their flow/header[3] counterpart.

A detailed description of the different type of captured/collected traffic is given in Table 1. Under Secure web label, we have HTTPS traffic mainly generated while browsing through Firefox and Chrome. Any browser related activities such as voice call using facebook or streaming through Youtube do not belong

[3] dependent on network latency, jitter, packet loss etc. that are less likely to remain across heterogeneous network.

Table 1. Number of payloads in the dataset employed

Application	Source	Type	Number of Payloads	Label/Class
HTTPS	ISCX Dataset [7]	Secure web	1814708	Encrypted
Skype	TStat Dataset [8]	VOIP	1945827	Encrypted
DTLS	Wireshark Sample Capture [9]	TLS	6084	Encrypted
LDAP	Wireshark Sample Capture	TLS	73655	Encrypted
SSL	ISCX Dataset	Tunnel	1228889	Encrypted
Youtube	ISCX Dataset	Streaming	194927	Encrypted
HTTP	ISCX Dataset	Weblogs	323158	Unencrypted
DecPhone	Wireshark Sample Capture	VOIP	25868	Unencrypted
Unencrypted Video	ISCX Dataset	Streaming	187196	Unencrypted
DNS	ISCX Dataset	Internet Service	37972	Unencrypted
NetBioS	ISCX Dataset	Network	62224	Unencrypted
SMTP	ISCX Dataset	Mail	4030	Unencrypted
		Total Encrypted payloads		5264090
		Total Unencrypted payloads		640448

to this label. The Voice over IP (VOIP) includes traffic generated by voice application such Skype, DecPhone. Streaming data, a continuous and firm stream of data, includes data from Youtube, Vimeo. Unencrypted traffic payloads were collected mainly from 3 type of sources, such as, streaming video, VOIP, and weblogs (HTTP).

jNetPcap [6], an open-source java wrapper for libpcap library native calls, was used to extract payloads from network packet (pcap files). Note application layer payload and transport layer payload were separated. For our experiment, only application layer payloads were considered in order to avoid noises with encrypted payloads. Because in most of the cases, protocols' payload encryption is done in the application layer i.e., SSL, HTTPS etc. For instance, length of application layer payload/transport layer payload (in bytes) for some packets in our experiments were 1133/1338, 3663/4036, 2280/2700 etc. Note that application layer payloads are smaller than that of transport layer.

For training data set we discard any payloads smaller than 20 Bytes (including noise) mainly due to bias such small payloads introduce into the model causing highly unreliable result. Finally, we end up with a large dataset consisting of encrypted payloads from 5264090 packets and unencrypted payloads from 640448 packets as shown in Table 1.

4 Experiment and Results

4.1 Feature Selection

Features for the classifier were chosen from entropy of the entire payloads, n-length Word or sliding window entropy, entropy of the encoded payload, and

Table 2. Entropy Based Features Employed

Encoded payload	Abbreviation	Binary payload	Abbreviation
Entire-file entropy	e_ENC	Entire-file entropy	e_BIN
n-length Word entropy	n_ENC	n-length Word entropy	n_BIN
		Mean of n-length Word Bintropy	$n\_\widehat{\texttt{BIN}}\_m$
		Std. Deviation of n-length Word Bintropy	$n\_\widehat{\texttt{BIN}}\_s$

BiEntropy: a logarithmic weighted average of the Shannon binary entropy. The following three algorithms were considered in calculation of entropy:

1. Entropy of the entire file where conventional Shannon's entropy (in [4]) is used for measurement. In case of a binary file, the entropy is measured in bits where the *symbol* is either 0, or 1. However, in case of a HEX-encoded file, there are 16 distinct symbols: [0–9], [A-F].
2. Entropy of n-length Word (where $n = 2$ to 64) or sliding-window technique [5]. It measures Shannon entropy over a sliding window of all the *word* tokens in a file. For instance, measuring byte entropy over a sliding window (where *word* size or window size $n = 8$) provides entropy value for each byte.
3. Mean and Std. deviation of n-length Word BiEntropy (in [33]) to detect the existence of repetitive pattern. First, calculate all the binary derivatives ($< n$) of a n-length word. If the last derivative is 0, word is *periodic* BiEntropy is 0 and vice versa.

Table 3. Selected Features by Genetic and C5imp

Method	Number of features	Selected Features
GF-SVM	8	e_ENC (HEX), e_BIN, 8_ENC (UTF-8), 8_BIN, $32\_\widehat{\texttt{BIN}}\_m$, $32\_\widehat{\texttt{BIN}}\_s$ $8\_\widehat{\texttt{BIN}}\_s$, $16\_\widehat{\texttt{BIN}}\_s$
C5imp	10	e_ENC (HEX), e_BIN, $24\_\widehat{\texttt{BIN}}\_m$, 12_BIN, 8_BIN, $12\_\widehat{\texttt{BIN}}\_m$, $32\_\widehat{\texttt{BIN}}\_m$, 8_ENC (UTF-8), $8\_\widehat{\texttt{BIN}}\_s$, $4\_\widehat{\texttt{BIN}}\_m$

While (1) and (2) are applied to *encoded* (HEX and UTF-8) payloads, *all* the algorithms are applied to the *binary* payloads as shown in Table 2. Note that all the entropy metrics were transformed by a logarithmic function since it has been observed to greatly increase machine learning accuracy. Resultant entropy were also normalized based on unity-based normalization before applying genetic algorithm.

Feature selection based on Genetic algorithm (in [34]) follows several steps: chromosome coding, initial population or feature chromosome, training Least Square SVM (LSSVM) classifier, fitness function, and terminating condition. For *chromosome coding*, we encode selected feature subset into a binary code[4]

[4] where 1 indicates the features to be selected and 0 not.

consisting of the total number of features. Regarding *initial population*, we choose some important features from our initial experiments (e.g., e_ENC (HEX), 32_$\widehat{\texttt{BIN}}$_m, 8_$\widehat{\texttt{BIN}}$_s etc.). LSSVM classifier helps to evaluate initial population (through classification accuracy) and hence the genetic algorithm to decide the effective features finally. The most important step in genetic algorithm is to construct an effective *fitness function* that evaluates a feature's fitness to survive. Chromosomes must be evaluated by fitness function. Our fitness function stems from weighting True Positive Rate (TP), False Positive Rate (FP) and the number of selected features $\texttt{N}(s)$.

$$\texttt{Fit}(s) = w_t \times \texttt{TP} + w_f \times \texttt{FP} + w_s \times \texttt{N}(s) \tag{1}$$

where w_t, w_f, and w_s be the weight values for $\texttt{TP}, \texttt{FP}$ and $\texttt{N}(s)$ respectively. In our experiment, we set $w_t = .5, w_f = .3$ and $w_s = .1$ to achieve high TP, low FP, and a small subset of selected features. In each iteration, chromosome with highest fitness value survives by which genetic operator creates new generation. The algorithm *terminates* when the maximum number of iteration takes place.

We start with the features in the Table 2 with varying the value of n. However, considering all possible combinations of features for calculating *detection rate* would require a significant computational cost while dealing with large dataset. That is why, we make several small groups i.e., features based on Bintropy, features based on entire file etc. and run genetic algorithm on each individual group. Apart from Genetic algorithm, we employ C5imp, a variable importance measures for C5.0 models with metric 'splits'[5] with 100 different trials to maximize the C5.0 classifier accuracy.

Table 3 demonstrates the number and labels of important/ optimal features selected by Genetic and C5imp feature selection algorithms. Results show that C5-imp mainly weights on n-length Bintropy based features while both of them consider encoded/binary payload based features.

4.2 Result Discussion

In this section, we discuss the detailed experiments and results of validation on the traced datasets. First, a set of statistical features based on encoding and entropy of the packets' payload has been studied for the experiment. Following the model of Zhao et al. (in [34]) where Genetic algorithm and LSSVM are used to were used to develop a detection model with effective feature selection. We use the same model but to detect encrypted/unencrypted traffic through network.

First, we select optimized number of features for encrypted and unencrypted traffic payload by creating feature chromosomes and evaluating them with the highest classification accuracy. Second, training LSSVM with the training data, that is, support vectors will be created based on training traffic traces. Finally, classifying the traffic data into two class encrypted and unencrypted. Nonetheless, in order to identify an ideal model for encrypted traffic classification and

[5] To calculate the percentage of splits associated with each predictor.

compare the robustness of the models with respect to selected features Genetic-LSSVM, LibSVM, and C5.0 algorithms were employed.

This study has employed a LSSVM to distinguish 2 classes (encrypted and unencrypted) of traffic data. Our experiments took place on a Ubuntu platform having configuration Intel core i7 3.6GHz, 32 GB RAM. Feature subset has been selected based on Genetic algorithm and C5imp in terms of ensuring the highest classification accuracy, TP and the lowest FP. The use of top ranked features (8 features out of more than 110 features) for classification give a drastic change in TP and FP values (Table 2). LS-SVMlab toolbox in [35] is used to implement LSSVM and modeling the classifier. LibSVM library available in weka [27] is used to run SVM algorithm with all features. RStudio [28], an integrated development environment (IDE) for R, is used for decision tree based model C5.0.

Table 4 shows the numeric result of the proposed Genetic-LSSVM in comparison to the all features selected SVM model. In fact, proposed model selects the best feature subset and then final results of the classification with the selected features are shown in the table. As the result shows, there is a significant change in both the encrypted and unencrypted class performance using Genetic-LSSSVM, where DR has reached 97 % while it was approx. 87 % using SVM with all possible features. FPR results are also noticeable as it has a 28 % reduction.

In order to compare our model, we apply decision tree and rule based model C5.0 and find 98.1 % accuracy from 10 selected features with minimal error (0.9 %). Note that the accuracy of C5.0 decision tree algorithm depends on the highest Information Gain of the features [26].

High detection rate can be better explained by (i) considering only application layer payloads in case of TCP payloads, (ii) the nature of the type of data, e.g., Skype or HTTPS payloads are huge and might be partially dominating the dataset (iii) dataset is partly biased containing payloads from limited number of applications. Results may fluctuate with much more diverse set of traffic from many applications/protocols.

4.3 Evaluation and Validation

In order to validate the selected features (e.g., word size, encoding scheme etc.), experiments have been done in two steps: (i) using randomly selected plain-text and their corresponding encoded-text and cipher-text, and (ii) using clustering algorithm as an alternative approach to classify traffic and to observe whether cluster analysis can identify encrypted or unencrypted traffic effectively using only the selected features by genetic algorithm. Note that, there is a sharp disparity between the entropies of plain-text and cipher-text. However, Encoded-text's entropy is very close to that of Encrypted-text.

Bintropy based features, that is, a weighted average of the Shannon entropy, show significant differences between the encrypted and unencrypted (but encoded) binary payloads. Especially features from Std. deviation ($32\_\widehat{\texttt{BIN}}\_s$) was promising in most of the case. On the other hand, Encoded-text's entropy, whether from *entire* file or from n-length *word* (e_ENC, 8_ENC), demonstrate

keen difference to the encrypted. Nonetheless, our feature selection algorithm chooses HEX encoding for *entire* file and UTF-8 encoding for n-length *word*. Although n-length word entropies (8_ENC and 8_BIN) indicate sharp divergence in this example, for a large file (payloads) difference is not substantial. This experimental procedure has been applied to different size of text files and observed the entropies for the feature set selected by the genetic algorithm and C5imp.

We consider an unsupervised algorithm, namely K-Means to evaluate and compare them with previous used Genetic-LSSVM and C5.0 classification algorithms using the GF-SVM features. It consists of 2 stages: a model generation and a classifier generation. In the former stage, K-Means algorithm clusters training data to produce a set of clusters to be labelled. In the latter stage, this model is used to develop classifier that can label any traffic traces. K-Means clustering evaluated with K initially being 2 exhibits overall accuracy exceeding 96 % with 7 iterations. The overall accuracy was sharply declined to 54 % with 25 iterations when K was 3. This declining continued until we check for K is 10. Accuracy results for DBSCAN clustering is 89.7 % (inc. noise or unclustered 24 instances) with two input parameters: ϵ is 0.3 where at least minPt is 3. The additional 6 clusters found using 3 minPt were typically small clusters containing 3 to 31 instances. However we did not trial with different ϵ and/or minPt that may improve its overall accuracy.

In order to train each of the learning algorithms properly several runs were conducted with different parameter values. It helps to generate accurate model with highest gain and ensures the result is not biased or dominated but from statistically compelling trails.

Table 4. Accuracy using Genetic-LSSVM, traditional LibSVM, C5.0 algorithms

Model	Correctly	FPR		DR	
	Classified instances	Encrypted	Unencrypted	Encrypted	Unencrypted
Genetic-LSSVM	96.639 %	0.034	0.033	0.967	0.966
LibSVM (all features)	86.954 %	0.29	0.31	0.693	0.761
C5.0 (C5imp features)	98.10 %	-	-	-	-
K-Means (GF-SVM features)	95.90 %	-	-	-	-
DBSCAN (GF-SVM features)	89.70 %	-	-	-	-

5 Conclusion

Our primary motivation was to find the solution to detect encrypted traffic from the network traffic depending on entropy-based features. We investigated several interesting characteristics of traffic payloads related to encoding, *n-length* word of the raw byte distribution and observe the aftermath effect on entropy measurement.

However, we have several shortcomings though. First, although we try to accumulate packets from the popular encrypted protocols, the dataset we worked do not consist of each and every type of encrypted or unencrypted traffic. Aside from these, in some applications, first few payloads are partially encrypted or unencrypted, we could not consider that in this time. That is why, we do believe that still there are some noises in the experiment training dataset. Although our dataset does not contain every kind of encrypted or unencrypted application traces, effectiveness of different combination of entropy features in terms of getting more detection accuracy has been fully studied.

As a future work, we have plan to investigate our proposed model on a wider range of applications and heterogeneous datasets from various networks. We also aim at analyzing protocol's individual signature from entropy and cross-validate its consistency with clustering algorithms. Finally, we plan to apply this approach to reveal signatures for network intrusion detection system. That would make it easy to infer if the encrypted traffic is benign or something that should be investigated further.

Acknowledgements. This work was funded by Atlantic Canada Opportunity Agency (ACOA) through the Atlantic Innovation Fund (AIF) in cooperation with IBM Security division.

References

1. Callado, A., et al.: A Survey on Internet Traffic Identification. IEEE Commun. Surveys Tutorials, **11**(3), 37–52 (2009)
2. Alshammari, R., Nur Zincir-Heywood, A.: Can encrypted traffic be identified without port numbers. Computer networks **55**(6), 1326–1350 (2011)
3. Alshammari, R., Nur Zincir-Heywood, A.: Investigating two different approaches for encrypted traffic classification. Privacy, Security and Trust (2008)
4. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379–423 (1948)
5. Marsaglia, G., Zaman, A.: Monkey tests for random number generators. Comput. Math. Appl. **26**(9), 1–10 (1993)
6. jNetPcap, Open-source java library. http://jnetpcap.com
7. Datasets: Information Security Center of eXcellence (ISCX). www.unb.ca/research/iscx/dataset/
8. Tstat, Skype Testbed Traces. http://tstat.tlc.polito.it/traces-skype.shtml
9. Wireshark sample captures. http://wiki.wireshark.org/SampleCaptures
10. Schneider, P.: TCP/IP traffic Classification Based on port numbers. Division Of Applied Sciences, Cambridge, **2138** (1996)
11. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINK: multilevel traffic classification in the dark. In: SIGCOMM , Philadelphia, 21–26 August 2005
12. Moore, A.W., Zuev, D.: Internet traffic classification using bayesian analysis techniques. In: SIGMETRIC, Banff, 6–10 June 2005
13. Sen, S., Spatscheck, O., Wang, D.: Accurate, scalable in-network identification of P2P traffic using application signatures. In: WWW2005, USA (2004)
14. Zander, S., Nguyen, T., Armitage, G.: Automated traffic classification and application identification using machine learning. In: LCN, Australia (2005)

15. Gomes, J.V., et al.: Analysis of peer-to-peer traffic using a behavioural method based on entropy. In: Performance, Computing and Communications Conference (2008)
16. Bonfiglio, D., et al.: Revealing skype traffic: when randomness plays with you. In: Proceedings of the ACM SIGCOMM, pp. 37–48. ACM Press, USA (2007)
17. Smith, R., et al.: Deflating the big bang: fast and scalable deep packet inspection. In: ACM SIGCOMM , pp. 207–218. ACM Press, USA (2008)
18. Zhang, H., Papadopoulos, C., Massey, D.: Detecting encrypted botnet traffic. In: Computer Communications Workshops (INFOCOM Workshop). IEEE (2013)
19. Dorfinger, P., et al.: Entropy-based traffic filtering to support real-time Skype detection. In: Proceedings of the 6th International Wireless Communications and Mobile Computing Conference. ACM (2010)
20. Korczynski, M., Duda, A.: Markov chain fingerprinting to classify encrypted traffic. In: INFOCOM, Proceedings IEEE. IEEE (2014)
21. Sun, Q., et al.: Statistical identification of encrypted web browsing traffic. In: IEEE Symposium on Security and Privacy, Proceedings. IEEE (2002)
22. Weber, M., et al.: A toolkit for detecting and analyzing malicious software. In: 18th Annual Proceedings of Computer Security Applications Conference. IEEE (2002)
23. Lyda, R., Hamrock, J.: Using entropy analysis to find encrypted and packed malware. IEEE Secur. Privacy **2**, 40–45 (2007)
24. Olivain, J., Goubault-Larrecq, J.: Detecting subverted cryptographic protocols by entropy checking. Laboratoire Specification et Verification, ENS Cachan, France, Research Report LSV-06-13 (2006)
25. Wagner, A., Plattner, B.: Entropy based worm and anomaly detection in fast IP networks. In: WETICE 2005 Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, pp. 172–177. IEEE Computer Society, Washington, DC (2005)
26. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Francisco (1993). ISBN=1-55860-238-0
27. Hall, M., et al.: The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter **11**(1), 10–18 (2009)
28. RStudio, an integrated development environment (IDE)for R. http://www.rstudio.com
29. Sicker, D.C., Ohm, P., Grunwald, D.: Legal issues surrounding monitoring during network research, In: Proceeding 7th ACM SIGCOMM conference on Internet measurement, ser. IMC 2007, pp. 141–148. ACM, New York (2007)
30. Chung, J.Y., Park, B., Won, Y.J., Strassner, J., Hong, J.W.: Traffic classification based on flow similarity. In: Nunzi, G., Scoglio, C., Li, X. (eds.) IPOM 2009. LNCS, vol. 5843, pp. 65–77. Springer, Heidelberg (2009)
31. Keralapura, R., Nucci, A., Chuah, C.-N.: Self-learning peer-to-peer traffic classifier. In: Proceedings of 18th Internatonal Conference on Computer Communications and Networks, ICCCN. IEEE (2009)
32. TCPDUMP packet analizer. http://www.tcpdump.org
33. Croll, G.J.: BiEntropy-The Approximate Entropy of a Finite Binary String (2013). arXiv preprint arXiv:1305.0954
34. Zhao, M., et al.: Feature selection and parameter optimization for support vector machines: a new approach based on genetic algorithm with feature chromosomes. Expert Syst. Appl. **38**(5), 5197–5204 (2011)
35. LS-SVMlab toolbox. http://www.esat.kuleuven.ac.be/sista/lssvmlab

36. Erman, J., Arlitt, M., Mahanti, A.: Traffic classification using clustering algorithms. In: Proceedings of the SIGCOMM workshop on Mining network data. ACM (2006)
37. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)