Network Forensics of Partial SSL/TLS
Encrypted Traffic Classification Using
Clustering Algorithms

Meng-Da Wu        Stephen D. Wolthusen

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX
United Kingdom
{M.D.Wu, Stephen.Wolthusen}@rhul.ac.uk

August 9, 2008

Abstract

Machine learning tools have long been used in network traffic analysis, but their application to the network forensics domain and its specific issues has been limited thus far. We investigate the applicability of several common machine learning techniques to identify and classify partial encrypted traffic as may be encountered by forensic investigators confronted only with partial post-hoc traces. It is highly desirable to identify the types of applications and endpoints using such tunnels to facilitate further forensic investigation. In this paper, we therefore examine several clustering algorithms, namely DBSCAN (Density-Based Spatial Clustering of Application with Noise), K-means, and EM (Expectation-Maximization) with regard to their ability to classify encrypted partial traffic using inter-arrival time and TCP length information chosen for its predictive significance. Our experiments demonstrate promising classification results.

1 Introduction

Network traffic analysis has been the basis for a number of application areas, including network design, management, and particularly network security where it is crucial for intrusion detection and prevention. The latter area has therefore seen considerable research devoted to both capturing and subsequent processing and analysis steps.

One particular challenge faced in this is the use of encrypted connections and tunnels in a number of application areas. While it is highly desirable to secure legitimate network traffic in this way, it is not an unmitigated benefit as this generally implies that monitoring will be considerably less effective unless special precautions are taken and e.g. encryption gateways or host-based traffic capturing are used. Even when such mechanisms are available, however,
significant performance issues remain since the capturing and filtering mecha-
nism must provide the aggregate decryption bandwidth of all monitored traf-

However, the conditions under which network forensics must operate dif-
fer in serveral important ways from those found in intrusion detection and
prevention. As investigations will typically be post-hoc, the infrastructure of
encryption gateways or even host-based monitoring mechanisms cannot be
assumed. As a result, any traffic being encrypted by endpoints with key mate-
rial that is not readily accessible to investigators will be available in encrypted
form only. Given the limitations of using pre-shared keys, however, key mate-
rial will generally be negotiated dynamically in key agreement protocols and
hence not be available for subsequent investigation unless suitable precautions
have been taken.

Forensic investigations will therefore frequently be confronted with encrypted
traffic and must deduce as much as possible from this limited information.
Since an increasing amount of legitimate network traffic is also encrypted, this
requires discerning between legitimate and illegitimate traffic, both of which
may be encrypted using the same protocol such as SSL/TLS or, in some cases,
IPSec. Naive address and port number mapping to identify protocols is insuffi-
cient as particularly port numbers are routinely changed, and SSL/TLS tunel-
ing is typically effected through ports assigned to HTTP(S).

The remainder of this paper is organized as follows: Section 2 provides
a brief overview of the specimen tunneling protocols (SSL/TLS) analyzed to-
gether with a discussion of the machine learning algorithms used for classify-
ing traffic followed by a review of the related work in section 3.

Section 4 then briefly describes methodological issues and the design of
the experiment subsequently discussed in section 5. The mechanisms used to
identify individual protocols in encrypted tunnels are then described in section
6 before providing a discussion of results obtained in section 7 and an outlook
on ongoing and future work in section 8.

2 Background

The following section briefly reviews key features relevant for the analysis of
tunneled protocols in the SSL/TLS protocol suites in section 2.1 and also pro-
vides an overview of the different machine learning algorithms selected to per-
form the classification of application tunnel behavior.

2.1 The SSL / TLS Protocols

The SSL (Secure Sockets Layer) / TLS (Transport Layer Security) protocol suite
is situated above the transport (TCP) layer and provides a number of secu-
urity services, including traffic encryption, client-side and server-side authen-
tication, and message integrity, which can be combined in a flexible manner
to support differing application requirements [Res00]. While it was originally
developed to secure connections between web servers and browsers – which
is still the most popular application area for the protocol suite – it is now in-
creasingly used to provide security services for other applications and also to
provide general secure network connectivity and not restricted to a single protocol such as HTTP (Hypertext Transfer Protocol). Any upper-layer protocol or application relying on TCP for connection-oriented transport can integrate security services provided by SSL/TLS (e.g. FTP, POP, IMAP).

SSL/TLS is a multilayer protocol consisting of four separate components. The SSL Record Protocol formats and encapsulates messages from upper protocols before passing them to TCP layer. The SSL Handshake Protocol allows clients and servers to agree on a protocol version, to authenticate both sides, and to negotiate a cipher suite. The SSL Change Cipher Spec Protocol message is sent by either the client or the server to notify the receiving party that subsequent content will be protected by the newly negotiated CipherSpec and Keys. Finally, SSL Alert Protocol messages are generated when either party to the communication session encounters an error or intends to close a session.

The protocol suite is flexible in its configuration and choice of parameters, and it can be implemented by different requirement, such as server-side only authentication, mutual authentication, or (historically) even using the FORTEZZA handshake.

2.2 Machine Learning Algorithms

The problem of classifying data can be addressed in a number of ways; machine learning provides an attractive approach if training data is available and variations on a given pattern are to be recognized later without requiring manual intervention. Regardless of the concrete algorithms, such solutions usually consist of two processes. One is the classifier construction (learning) phase, and the another one is the actual classification or testing phase where the data sets for each application must be disjoint.

A further distinction can be drawn between supervised and unsupervised techniques. In the latter, clustering algorithms group patterns into different clusters according to feature similarities. This approach does not require pre-labeled data sets, and the goal of the algorithm is to extract a number of hidden clusters from an un-labeled data set. In the case of supervised classification, functions are defined when mapping from a previously labeled set of input data (feature) vectors onto a finite set of discrete class labels.

While no plaintext is available for ready classification in case of encrypted tunnels, we posit that sufficient and statistical features are available for inferring clusters and use unsupervised clustering algorithms. To this end, three unsupervised clustering algorithms were chosen to analyze the data sets which are not only encrypted but also incomplete (only capturing one portion of the whole traffic set) typical for forensic analysis. Other research paper regarding to these algorithms can be referred to [BTA+06, BTS06, MLC07, EAM06, BSAS05, EMAW07, MHLB04, EMA06, ZNA05]. The following provides a brief summary of the attributes of each chosen algorithm.

2.2.1 DBSCAN

The Density-Based Spatial Clustering of Application with Noise (DBSCAN) [EKSX96, Han05] algorithm was first proposed to cluster spatial databases by using density-oriented criteria. It is designed to discover clusters of arbitrary shape. It regards clusters as dense areas of objects which are separated by
cluster is considered as noise data. Density-reachability and density-connectivity are the two key concepts to form clusters. For implementing these two concepts, two global parameters are defined: \( Eps \), the maximum radius of the neighborhood; and \( MinPts \), giving minimum number if points in an \( Eps \)-neighborhood of that point.

Further definitions in the algorithm:

**Density Reachability** A point \( p \) is density reachable from point \( q \) if the following two conditions are satisfied:

1. the points are close enough to each other: distance\((p, q) < Eps\)
2. there are a sufficient number of points in a \( q \)-neighborhood

**Density Connectivity** A point \( p \) is density connected to a point \( q \) if there is a point \( o \) such that both, \( p \) and \( q \), are density reachable from \( o \).

**Core Point** A point with at least \( MinPts \) objects within a \( Eps \)-neighborhood.

**Border Point** A point on the border of a cluster

The DBSCAN algorithm can be sketched as follows:

1. Select a fixed but arbitrary point \( p \)
2. Retrieve all points density-reachable from \( p \) with respect to \( Eps \) and \( MinPts \).
3. If \( p \) is a core point, a cluster is formed.
4. If \( p \) is a border point, no points are density-reachable from \( p \) and DBSCAN visits the next point of the database.
5. Continue the process until all of the points have been processed.

### 2.2.2 K-Means

The \( k \)-means algorithm [HW79, Han05] is a partitioning algorithm. This kind of algorithm organizes all objects into \( k \) partitions where each partition represents a cluster. The \( k \)-means algorithm is the simplest one to perform clustering testing.

The algorithm takes the input parameter \( k \) and partitions a set of \( n \) objects into \( k \) clusters. Cluster similarity is measured with regard to the mean value of the objects in a cluster, which can be viewed as the cluster’s center. For each cluster, the clustering algorithm maximizes the homogeneity within the cluster by minimizing the square-error, i.e. the process iterates until the criterion function converges. Typically, the square-error criterion is used, defined as

\[
E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2
\]

where \( E \) is the sum of the square error for all objects in the data set; \( p \) is the point in space representing a given object; and \( m_i \) is the mean of cluster \( C_i \). For
each object of each cluster, the distance from the object to its cluster is squared, and the distances are summed. The k-means procedures are summarized as the following:

**Input** $k$, the number of clusters; $D$, a data set containing $n$ objects

**Output** A set of $k$ clusters

**Procedures**
1. Arbitrarily choose $k$ objects from $D$ as the initial cluster centers
2. Iterate the following until the clusters are stable:
   - (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster
   - update the cluster means, i.e. calculate the mean value of the objects for each cluster;

### 2.2.3 EM

The final algorithm considered here is the EM (Expectation-Maximization) algorithm [Han05, DLR77]. It is typically used to perform maximum likelihood estimation of parameters in probabilistic models. By using the EM clustering technique, each cluster can be presented as a parametric probability distribution.

It assumes that the entire data set can be treated as a mixture of underlying probability distributions, which is typically referred to as a component distribution. This technique can cluster data sets by using a finite mixture density model of $k$ probability distributions, where each distribution represents a cluster. It is aiming to find the best fit of the entire data by estimating the parameters of probability distributions. EM is a iterative refinement algorithm to estimate parameter by assigning each object to a cluster according to a weight representing the probability of membership. There are no strict boundaries between clusters, and new means are computed based on the specification of weighted measures. The algorithm is described as the following steps:

1. Provide an initial estimation of the parameter vector
2. Iteratively refine the parameters by using the following two steps:
   - **Expectation Step:** Assign each object $x_i$ to cluster $C_k$ with the probability
     $$P(x_i \in C_k) = p(C_k \mid x_i) = \frac{p(C_k)p(x_i \mid C_k)}{p(x_i)},$$
     where $p(xi \mid C_k) = N(m_k, Ek(x_i))$ follows the normal distribution around the mean, $m_k$ with expectation $E_k$. This step calculates the probability of cluster membership of object $x_i$ for each of the clusters. These probability are the expected cluster memberships for object $x_i$.
   - **Maximization Step:** Use the probability estimates from above to re-estimate the model parameters, i.e.
\[ m_k = \frac{1}{n} \sum_{i=1}^{n} x_i P(x_i \in C_k) \]

3 Related Work

In this paper, we examined three unsupervised clustering algorithms of machine learning techniques: DBSCAN (Density-Based Spatial Clustering of Application with Noise), K-means, and EM (Expectation-Maximization). These techniques have been applied to network traffic analysis domain in many literatures. K-means is the most straightforward clustering method in traffic classification [BTA +06, BTS06, MLC07, EAM06, BSAS05, EMAW07]. DBSCAN clustering technique was used to cluster network traffic for accurately identify applications [EAM06]. The EM applications in network traffic were examined and discussed in [MHLB04, EAM06, EMA06, ZNA05]. Further reference of clustering technique survey can refer to [XW05]. Other machine learning techniques also demonstrated promising results in network traffic analysis, such as Hidden Markov Model [WMM06, BTS06, BSAS05], Bayesian classifier [ZNA05], and K-Nearest Neighbor [WMM06]. On the specific analysis application of SSL/TLS tunnel traffic, [BLJ06, SSW +02] used statistics techniques to analyze HTTPS traffic. [WW07] exploited the information leakage of SSL/TLS sequential headers for classifying software implementations.

In the network forensics domain, research literatures are good reference when performing practical analysis. [RJ05] modeled network forensics mechanism, and [EPF06] proposed network forensic readiness development life cycle. Both of them help to provide ont only a network forensic sound methodology but also a comprehensive overview. [WD05] proposed to build evidence graphs for enhancing network forensics analysis. The network uncertainty could lead to forensic investigation errors and should be concerned. This uncertainty of network evidence were discussed in [Cas02], such as data corruption, loss, tampering, or errors. More practical concerns and comparisons of choosing network forensics tools and functions can be found in [Cas04]. Improvements of network evidence collection are discussed in [Nik06].

4 Methodology of applying clustering algorithms

We employed standard classification mechanisms to classify partial traffic, obtaining \( n \) cluster descriptions per algorithm, which are described as clustering centers in the following discussion. Following the learning phase, each algorithm processed a specific clustering composition.

In the testing phase, each individual partial traffic set is first clustered by the clustering algorithms, yielding a candidate clustering composition. This candidate composition is then classified (compared) to all other pre-defined clustering compositions trained in the learning phase.

Our classifier is based on a simple Euclidean distance metric for candidate clustering centers \( X_i \) and pre-defined clustering centers \( Y_i \) as
Table 1: Software Specimen using SSL/TLS

\[ Dist(X, Y) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (X_i - Y_i)^2} \]

where \( Y_i \) is given by one of the algorithms described above. The \( Dist \) is computed by \( n \) clustering centers produced by different clustering algorithms and parameters.

5 Application classification using partial encrypted traffic

In the following section, the experimental methodology is briefly described, followed by the results of the experiments.

5.1 Experiment Scenario

Our mechanism employs partial traffic to classify network protocols which is encrypted by SSL/TLS. This is owing to the expectation that in a forensic application, it will be unlikely to have full traces available. We therefore chose 1000 continuous IP packets between two specific ports on a pair of machines as our defined partial traffic set. In the feature space, only the inter-arrival time feature is extracted and tested in this section.

There are 2 hosts in our scenario, and we extracted all parameters of their operation systems for the examining purpose. These parameters are highly critical because slightly changing OS parameters, such as TCP/IP stack, could lead to different traffic patterns. In such scenario, it is our task of network forensic investigations to classify network protocols by using partial traffic only.

5.2 Data Set Preparation

As this is a preliminary experiment, each protocol’s behavior presented here is extracted from one specific software implementation. Three protocols are chosen for testing our mechanism. The overall data variance of inter-arrival time feature in our lab is: minimum, 0.000001; maximum, 1.023213; mean, 0.003838. The data variance is based on such experiment scenario that the traffic we produced gets through a intermediary hub device where we collected all our traffic. The permutations of software package specimens used in the classification experiment are detailed in table 1.
Table 1 is the overall software\textsuperscript{1} information used for traffic generation. The operation system of both server and client side is Windows XP Pro SP2 with default TCP/IP parameters, and the SSL/TLS tunnel software is Barracuda HTTPS \textsuperscript{2}.

There are two data sets prepared for performing our mechanism: learning data set and testing data set. Learning data set is used to build the cluster composition of each specific software. The testing data set is for evaluating the accuracy of our classification mechanism. In real forensic cases, investigators can produce the learning data sets as many as necessary to guarantee sufficient data for forming reliable cluster compositions. In our experiments, 30 individual traffic sets as learning data are collected and promising classification results come out in later stage.

Both of the learning and testing data sets include 30 individual traffic sets for testing our mechanism. Parameters of the operation systems are the same during the data collection period, and they can be extracted from target machines by using computer forensics software, such as EnCase \textsuperscript{3}. Depending on these known parameters, we can design our mechanism without building a table referring to different pattern descriptions based on different parameter settings. Such table can be set up in a more complicated case investigation.

5.3 Learning Phase

According to our methodology, the core concept of learning phase is to compute the clustering composition of each software. Each clustering composition includes \( n \) clustering centers(clustering descriptions). In this phase, firstly we used unsupervised clustering algorithms to generate \( n \) clustering centers. We then computed the means of clustering centers from 30 training data sets. Afterward, these means are used to form clustering compositions. Such clustering compositions are used as classifier information in later stage of traffic classification.

5.4 Testing Phase and Results

Every partial traffic set (1000 continuous IP packets) in the testing phase is clustered and processed to form a candidate clustering composition. Each candidate composition is then classified (compared) to all other predefined clustering compositions trained in the learning phase. All experimental results are tested by \( R \) software \textsuperscript{4}, including DBSCAN, K-means, and Mclust[FR06] packages. Classification results are presented in the following sections:

\textsuperscript{1}Further information of our testing software:

\textsuperscript{2}See http://barracudaserver.com/products/BarracudaDrive/HttpsTunnel.lsp for details.

\textsuperscript{3}See http://www.guidancesoftware.com for details.

\textsuperscript{4}See http://www.r-project.org/ for details.
5.4.1 DBSCAN

For performing this clustering algorithm, two global parameters are required: *MinPts* and *Eps*. These two parameters can vary in a great scale, but we demonstrated DBSCAN’s strong capability in solving our defined problem by only choosing few parameter combinations. We chose the following parameter combinations of *MinPts/Eps*: 50/0.00005, 80/0.00008, and 110/0.00011. All of them have the same overall 99% accuracy in traffic classification.

The experimental results are generated by randomly choosing parameter combinations for DBSCAN. However, we did not really accept any parameter combinations, and we filtered them by an important factor: *we accept parameters only when they can generate stable clustering numbers*. In our observations, some parameter combinations generate unstable clustering numbers that could lead to difficulties in accurately describing data patterns. Unstable clustering numbers mean unreliable clustering separations. DBSCAN with unstable clustering numbers can not properly perform clustering description jobs, and we have to dismiss this kind of parameter combinations.

5.4.2 K-means

The K-means algorithm has only one input parameter: *K*. This input parameter presents how many clusters will be computed by its mathematical model. We tested the parameter from 2 to 100, and these experimental results are pre-
sented in Figure 1.

In this test, the overall average accuracy is 63%, and the best accuracy among all parameters is 79%. In the initial parameter 2, the accuracy is just 8%. After the parameter 2, the accuracy jumps to 58%. Most points of accuracy are in the range of 50%-75%. Even though the optimum accuracy can reach 79%, we can still see an unstable accuracy within all parameters. However, by separately observing each software’s classification results, the accuracy of a software can reach 97% after parameter 5. It means K-means can work very well in describing a software but fail to cluster the other two software behaviors. We could exploit this feature to develop another statistical mechanism for accurately identifying a software, but it is out of the scope of this paper. The purpose of this paper is to find a overall optimum algorithm in solving general partial traffic classification problem.

5.4.3 EM

In the EM algorithm, only one parameter $G$ is required to perform its mathematics model. $G$ presents how many clusters will be computed based on probability distributions. This part, we tested the parameter $G$ from 2 to 30. All experimental results can be found in Figure 2 (box-and-whisker plot). The best accuracy is parameter 19 with 91%. After the parameter 24, the EM capability in describing clustering goes down sharply. From the Figure 2, we can identify that the parameter range between 18 to 22 can generate better average accuracy of 89%. These data are good reference when performing real case investigations.

6 Protocol identification under SSL/TLS tunnel technology using DBSCAN algorithm

In section 5, we performed three different clustering algorithms to classify protocols, including FTP, real time streaming protocol, and remote framebuffer protocol. The strength of using DBSCAN to identify protocol behaviors were preliminary confirmed in section 5, but we used only specific software pair of both client and server sides with single feature. More experiments should be performed to estimate how DBSCAN can help forensic investigators to identify protocol patterns. Based on DBSCAN, we performed more experiments and added more features in this section to present the application of using DBSCAN clustering algorithms and conditional probability to identify protocol behaviors.

6.1 Experiment set up

In this section, we tested three popular network protocols, including File Transfer Protocol (FTP), Hypertext Transmission Protocol (HTTP), Real Time Streaming Protocol, and one malicious HTTP performed by Agobot\textsuperscript{5}. All experimental data is produced and collected as the same mechanism described in section

5. All traffic is encrypted by SSL/TLS technology and only partial traffic is captured for testing our mechanism. Table 2 is the overall software\textsuperscript{6} information used for traffic generation.

We used more combinations of both client and sever sides on each protocol to demonstrate our mechanism’s capability for dealing with the diversity of protocol behaviors. For each protocol, we chose three different client applications with the same server to have representative protocol traffic sets. In our experiment, we have full control of Agobot, and we commanded our bot to download files from our web server by using HTTP mechanism. Such malicious HTTP traffic is collected during file transfer duration.

We randomly chose one specific software implementation of each protocol to produce 30 learning data sets.

As we use three different client applications communicating with the same server, 90 testing data sets are collected on each protocol. In the malicious traffic set, only 30 learning data sets and 30 testing data sets has been collected because we only used Agobot as our client.

\textsuperscript{6}Further information of our testing software:
<table>
<thead>
<tr>
<th>Software Category</th>
<th>Server Side(software version)</th>
<th>Client Side(software version)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) File Transfer Protocol</td>
<td>TitanFTP (5.34)</td>
<td>FTP Voyager (15.0.0.1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Classic FTP (1.02)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EFTP (3.3.1.145)</td>
</tr>
<tr>
<td>(B) Real Time Streaming Protocol</td>
<td>VLC Media player (0.8.6c)</td>
<td>Windows Media Player (9.00.00.3250)</td>
</tr>
<tr>
<td>(C) HTTP</td>
<td>Apache (1.3.34) on Linux (2.6.13.2)</td>
<td>VLC Media player (0.8.6c)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kantaris (0.3.7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mozilla Firefox (2.0.0.14)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Opera (9.27)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IE (6.0)</td>
</tr>
<tr>
<td>(D) Malicious HTTP</td>
<td>Apache (1.3.34) on Linux (2.6.13.2)</td>
<td>Agobot</td>
</tr>
</tbody>
</table>

Table 2: Software Specimens using SSL/TLS

6.2 Feature collection

Only *inter-arrival time* feature is extracted in section 5. This section, we enrich our feature space by adding another 2 TCP level features. These new TCP level features are calculated from the inter-arrival time feature which is produced by the same mechanism introduced in section 5. As DBSCAN can form several clustering groups by using inter-arrival time information, these separated groups can be used as conditions to produce our TCP level conditional probability features.

The first feature is to use the method in section 5: DBSCAN is performed by inter-arrival time information. Afterward, clustering results are presented as separated timing groups. The second feature is to extract the conditional probability of TCP length information. Such TCP information are pre-grouped by using clustering criterion from feature 1.

The second feature of conditional probability in TCP length information can lead to our third feature used in this section. There are several clustering groups of TCP length information found by calculating conditional probability feature, and it can be used as our third feature information. We chose the highest and second frequent group of TCP length information as our third feature. In some cases, however, only one TCP length group can be found. We have to dismiss TCP length information which appears just randomly and is not strong enough to form a group. We set up a threshold of 5% proportion of overall TCP length points for choosing TCP length group.

6.3 Experimental results

We use Euclidean distance metric for classifying our testing data sets. One inter-arrival time feature and two TCP length information features are used to calculate the overall feature distance. In such distance metric, a promising 97.5% accuracy is presented from our experiments. Another interesting result should be pointed out that DBSCAN not only classified traffic protocol but also detected protocol violation. HTTP performed by malicious bot, named Agobot, has distinct clustering information compared to normal HTTP browsing traffic.
7 Results and Discussions

All the experimental results are from our network forensic lab environment, and therefore this paper should be treated as a preliminary survey of clustering algorithms in helping network forensic investigations. Protocol traffic coming from different software implementations is tested to explore the capability of clustering algorithms in solving partial traffic identification problem.

In section 5, three clustering algorithms are applied to identify three protocol implementations. The EM algorithm can have a overall 89% accuracy within the specific parameter rage. Thus, EM could be reliable to cluster inter-arrival time information after precisely experimental testing and setting. However, it would need more refinement for helping classification of protocol applications. Our K-means algorithm is a very basic version of implementation and does not deal with noise data. From the testing results, K-means failed to solve the partial traffic identification problem. By comparing the other 2 algorithms, DBSCAN presented highly satisfied capability in discovering traffic patterns. DBSCAN demonstrated an amazing 99% accuracy of software classification in section 5.

The section 6 is to explore DBSCAN’s application in partial traffic identification problem by adding more features. The advantage of DBSCAN is to extract density-based clustering information, and it works very well in discovering clustering information in our experiments. In section 6, it successfully extracted clusters of inter-arrival time and TCP length features. Moreover, the density-based algorithm can also automatically dismiss other noise data which is not within the clustering areas. The high accurate results show that good clustering descriptions can be formed by DBSCAN when choosing appropriate features. Such experimental results lead to the concept that network protocol implementation is a reliable subject of applying density-based clustering algorithm.

We used one specific implementation of SSL/TLS software as our tunnel technology in section 5 and 6, and all later experimental results should be considered in such circumstance. The purpose of this paper is to explore appropriate tools for extracting traffic patterns of communication behaviors. The software of SSL/TLS technology is implemented by its own specifications, and such distinct specifications are presented as different traffic patterns. Based on our experimental testing in section 5 and 6, network evidence examiner can rely on clustering algorithm, such as DBSCAN, for dealing with certain cases when highly sophisticated techniques are needed.

8 Conclusion and Future Work

Machine learning techniques have long been the subject of research in the network traffic analysis domain, but network forensics, particularly for encrypted traffic have specific requirements and have thus far not been investigated extensively. The main applications of network analysis are to classify traffic protocols by using full traffic data set with multiple features. Most literature in the domain aims to improve the accuracy or performance for such general traffic classification problems. Motivated by these papers, we believe machine learning applications are promising tools to enhance network forensics inves-
tigations with their specific application requirements such as having only partial data at its disposal along with noisy data sets. The use of unsupervised machine learning applications represent an efficient mechanism for classifying network traffic in such a way as to allow further investigation, potentially at the host level. To this end, we explored clustering algorithms in network forensics applications and, following promising initial results reported in this paper, intend to investigate further refinement of encrypted network traffic behavior, particularly in more noisy environments than those found in the laboratory environment used for the experiments reported here. Similarly, as the training data sets were generated in the same environment as the clustering data, network uncertainties can be assumed to be quite similar — this is not necessarily the case for normal forensic investigations. We are therefore investigating the influence of network uncertainty issues in further work, particularly fragmentation, fragment ordering, fragmentation timeouts, header errors, checksum errors, and duplicate packets at the IP level and duplicate segments, partially duplicate segments, segments arriving before or after the receive window, duplicate acknowledgments, and out-of-order segments for the TCP layer. At the algorithmic level, we will conduct further research on interactions between protocol layers and multivariate analysis of packet information.

References


