# Revealing Encrypted WebRTC Traffic via Machine Learning Tools

Mario Di Mauro and Maurizio Longo
*University of Salerno, Via Giovanni Paolo II 132, 84084, Fisciano (SA), Italy*

Keywords: Encrypted Real-Time Traffic, WebRTC, DTLS, Decision Trees, Weka, Machine Learning.

Abstract: The detection of encrypted real-time traffic, both streaming and conversational, is an increasingly important issue for agencies in charge of lawful interception. Aside from well established technologies used in real-time communication (e.g. Skype, Facetime, Lync etc.) a new one is recently spreading: Web Real-Time Communication (WebRTC), which, with the support of a robust encryption method such as DTLS, offers capabilities for encrypted voice and video without the need of installing a specific application but using a common browser, like Chrome, Firefox or Opera. Encrypted WebRTC traffic cannot be recognized through methods of semantic recognition since it does not exhibit a discernible sequence of information pieces and hence statistical recognition methods are called for. In this paper we propose and evaluate a decision theory based system allowing to recognize encrypted WebRTC traffic by means of an open-source machine learning environment: Weka. Besides, a reasoned comparison among some of the most credited algorithms (J48, Simple Cart, Naïve Bayes, Random Forests) in the field of decision systems has been carried out, indicating the prevalence of Random Forests.

## 1 INTRODUCTION

From a network operator perspective, the detection and classification of data flows over the Internet, is becoming a progressively challenging issue. In fact traditional methods, used by Internet Service Providers as admission criteria, rely on a list of registered and "well known" TCP/UDP port numbers provided by the Internet Assigned Numbers Authority (IANA), but are ineffective in detecting those network applications that randomly and dynamically select the ports exposing their services to the Internet (Freire, 2008). To obviate the random port option, one could exploit the recurrence of some specific patterns inside data sessions at the application layer which constitute a sort of "signature" of the targeted application (Song, 2008). Such approaches, also known as packet inspection methods, require that the packet's payload could be accessed and analyzed, thus are hardly applicable if the content is concealed by some form of encryption. Even in the plaintext, some data traffic could be not labeled by any identifier or, sometimes, such identifier may vary unpredictably, e.g. due to a different used application, thus making the pattern-based analysis unproductive.

The third way remaining after port number filtering and packet inspection is the traffic classification based on statistical features. In this paper we propose and assess a traffic classifier based on statistical pattern recognition able to detect at run-time an encrypted WebRTC session. WebRTC is a new standard that lets HTML5 compliant browsers to communicate in real-time using a peer-to-peer architecture (Loreto, 2014). This is a break-through in the application world since it enables web developers to build real-time multimedia applications with no need for proprietary plug-ins. The paper is organized as follows: in Section 2 a review of related contributions is presented; in Section 3 some details on the WebRTC standard are provided; in Section 4 the Waikato Environment for Knowledge Analysis (WEKA) is introduced with the relevant classification algorithms: J48, SimpleCART, Naïve Bayes and Random Forests; in Section 5 the proposed testbed is outlined along with the included software modules; in Section 6 some experimental results are presented and in Section 7 the main conclusions are drawn.

## 2 RELATED WORK

In recent literature, several network traffic

Di Mauro M. and Longo M..
Revealing Encrypted WebRTC Traffic via Machine Learning Tools.
DOI: 10.5220/0005542202590266
In *Proceedings of the 12th International Conference on Security and Cryptography* (SECRYPT-2015), pages 259-266
ISBN: 978-989-758-117-5

recognition and classification methods based on statistical features and on the use of a supervised machine learning approach have been introduced. Some such methods were proposed to classify various Internet applications: in (Xusheng, 2008) an approach to recognize P2P traffic (amongst BitTorrent, PPLive, Skype and MSN Messenger) based on the Support Vector Machine (SVM) is suggested; Bayesian analyses, implemented with the WEKA environment are considered in (Moore, 2005) and (Rahdari, 2012); a packet-level traffic classification approach based on Hidden Markov Model (HMM) is presented in (Dainotti, 2008).

In the supervised learning model, where the classes are known and defined beforehand, the main task is to infer the value of unknown parameters starting from some labeled training data. This approach is tipically divided in two phases (Nguyen, 2008):

- Training phase: the available dataset, or training dataset, is labeled by analysis or by construction and a classification model is built.
- Testing (or running) phase: the classification model is used to classify new unknown instances according to specified decision criteria.

The performances of the supervised network classifiers are very sensitive to the quality of the training dataset. Unluckily, perfect training datasets are difficult to obtain in practice since individual packets must be labeled by manual inspection in a controlled experimental environment (Lin, 2009). In our development we used a supervised approach dividing the data flows in two classes: *i)* the targeted WebRTC class consisting of encrypted real-time traffic generated using a browser supporting WebRTC; *ii)* the Normal class including all other types of data traffic like HTTP, FTP etc. In (Di Mauro, 2014) a similar approach for detection of Skype traffic was adopted.

## 3 WEBRTC OUTLINE

WebRTC is a new standard developed through an industrial effort to extend the web browsing model in such a way that browsers are able to directly exchange real-time media with other browsers in a peer-to-peer fashion. The World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) are jointly defining the WebRTC APIs enabling a web application running on any device, through secure access to input peripherals such as webcams and microphones, to exchange real-time media and data with a remote party. WebRTC extends the client-server semantics by introducing a peer-to-peer communication paradigm between browsers drawing inspiration from the so-called SIP Trapezoid (Rosenberg, 2002) as shown in Fig. 1. In this model, signaling messages are used to set up and terminate communications; these messages are carried by the HTTP or WebSocket protocol via web servers that can modify, translate, or manage them as needed. It is worth noting that while the WebRTC standard has been planned to fully specify the media plane, the signaling between browser and server is not standardized because it is considered as part of the application, in such a way that different applications can use different signaling protocols like Session Initiation Protocol (SIP) or eXtensible Messaging and Presence Protocol (XMPP). For the purpose of the present work it is important to outline the native security features of WebRTC. In fact the WebRTC stack includes the Datagram Transport Layer Security (DTLS) (Rescorla, 2006), a protocol designed to prevent eavesdropping and information tampering by encrypting real-time data (from microphone, camera or chat window) and offering a key and association

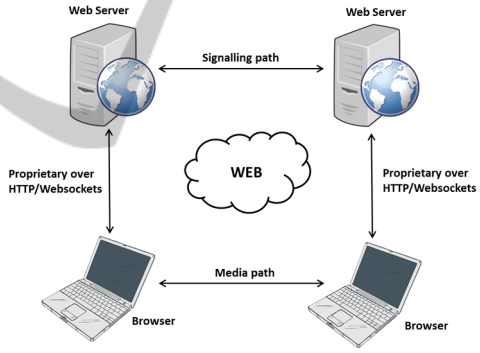


Figure 1: The WebRTC Model.

management service to Secure Real-time Transport Protocol (SRTP) (Baugher, 2004) as shown in Fig. 2. In itself, such service does not imply encryption of SRTP header, thus exposing to some deep packet inspection procedures. To counter this residual security flaw, several proposals are now available such as (Lennox, 2013). According to a worst case concept, we assume that the SRTP header is encrypted too. The end-to-end encryption between remote peers is managed on behalf of a TURN server (Mahy, 2010) enabled just to parse the UDP layer of a WebRTC packet without understanding nor modifying the application data layer. The Session Traversal Utilities for NAT (STUN) protocol (Rosenberg, 2008) instead, allows a host application to discover the presence of a network address translator and, in case, to obtain the

allocated public IP and port tuple for the current connection. WebRTC technology can be integrated within SIP-based systems offering new interesting possibilities. In (Zeidan, 2014) a videoconferencing architecture which allows legacy SIP user agents and modern SIP WebRTC clients to interoperate. In (Shi, 2013) a real-time chatting tool based on HTML5 and WebRTC is proposed; it includes a basic information management module, a chat module allowing text communication by means of a dedicated server and Voice/Video sessions through a point-to-point connection between browsers.

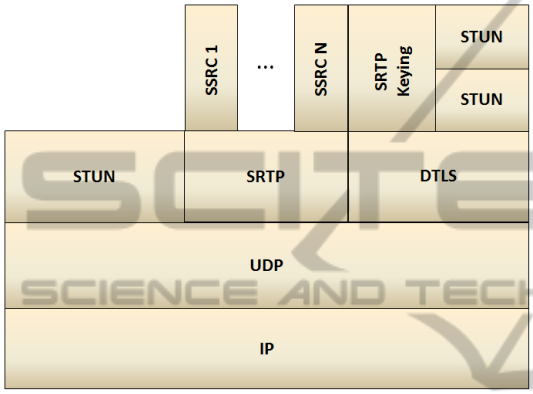


Figure 2: The WebRTC Protocol Stack.

# 4 WEKA: A MACHINE LEARNING TOOL

WEKA (Witten, 2005) is an open source package developed at the University of Waikato in New Zealand providing implementation of a quite large set of machine learning algorithms. It includes a collection of visual tools and algorithms for data analysis and predictive modeling as well as a set of methods for standard data analysis problems as regression, classification, clustering, association rule mining and attribute selection.

Weka can generally handle two types of file formats: CSV (Comma Separated Values) and ARFF (Attribute Relationship File Format), a native Weka format. The package is completed with a set of Java API in order to ease the task of software developers. In the testbed realized for the present work, we have implemented, based on said Java interfaces, some software modules to automatically extract the features characterizing encrypted WebRTC traffic. In order to build a statistically significant training dataset, a substantial amount of WebRTC and Normal traffic has been produced in the form of *pcap* traces, in a controlled setting, and then converted to CSV format. The WebRTC training data is a collection of 10 real time audio/video flows (about 10 minutes per session), globally worth 150 Mbytes; the Normal training data is a collection of various traffic segments like HTTP, FTP, etc., again globally worth 150 Mbytes.

## 4.1 The Proposed Algorithms

For the two-class (*WebRTC*, *Normal*) problem at hand, four algorithms have been actually tested: J48, SimpleCART, Naïve Bayes and Random Forests, all of them working with the same datasets both in the training and in the running phase, and using the same class attributes, namely

- *Proto*: Transport Level protocol type (TCP,UDP);
- *MinSLgt*, *MaxSLgt*, *MinRLgt*, *MaxRLgt*: minimum and maximum lengths of sent and of received packets
- *MinIAT*, *MaxIAT*: minimum and maximum inter-arrival times
- *StdSLgt*, *StdRLgt*: standard deviations of sent and of received packet lengths
- *StdIAT*: standard deviation of inter-arrival times

### 4.1.1 J48

The J48 (Meera Ghandi, 2010) algorithm represents the Weka implementation of Quinlan C4.5 algorithm (Quinlan, 1993) to generate decision trees for classification starting from a set of labeled training data. A decision tree is a tree data structure wherein a *node* indicates a test on one of the attributes and a *leaf* specifies a class value. The test on a continuous attribute $A$ has two possible outcomes: $A \leq t$ and $A > t$ where $t$ is a preset threshold (Ruggieri, 2002). The algorithm builds the decision tree adopting a *divide and conquer* strategy as summarized by the following general pseudo-code (Kotsiantis, 2007):

1. Check for base cases
2. For each attribute $X$ compute the Information Gain: *Gain (S,A)*
3. Letting *X-best* the attribute with the highest gain, create a decision node that splits on *X-best*
4. Recurse on the sublists obtained by splitting on *X-best* and add those nodes as children

The Information Gain of an attribute $A$ relative to a collection of training samples $S$ is defined as the reduction in entropy caused by partitioning $S$

according to such attribute, namely

$$Gain(S,A) = H(S) - \sum_{values(A)} \frac{|S_v|}{|S|} H(S_v) \quad (1)$$

where:

- $S_v$ is the subset of $S$ for which attribute $A$ has value $v$;
- $H(S)$, $H(S_v)$ are entropies;
- $|S|$, $|S_v|$ are cardinalities of $S$ and $S_v$ respectively;

It is well known that the entropy of a binary experiment, as in our case, lies in [0,1], with 0 attained if only one outcome is actually possible and 1 if every outcome is equally possible.

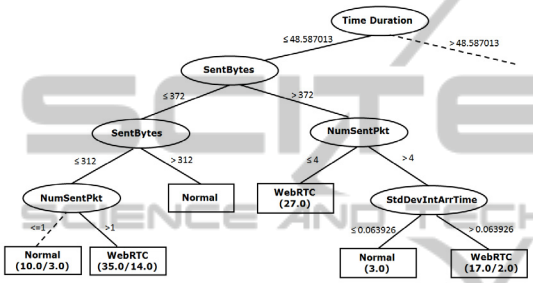


Figure 3: A subset of the decision tree for encrypted WebRTC traffic.

The Information Gain of an attribute can be interpreted as the reduction of entropy when the sample space is partitioned according to that attribute. It suggests that a criterion to iteratively build a classification tree is to split at a node corresponding to an attribute $A$ that maximizes the Information Gain. Once specialized for the classes and attributes of the problem at hand, J48 produces a decision tree as shown (in part) in Fig. 3. Consider the notation inside a leaf: for example, *WebRTC* (17.0/2.0) means that 17 instances of *WebRTC* reached the leaf and 2 out of them were incorrectly classified. The tree construction may be followed by an additional step called *pessimistic pruning*, a top-down method allowing to simplify the decision tree by substituting a single leaf in place of a whole sub-tree when a specific irrelevance condition is met, as suggested in (Kuhn, 2013).

### 4.1.2 Simple Cart

Simple CART algorithm arises from the CART (Classification And Regression Trees) method (Breiman, 1984), adapted to handle both discrete and continuous attributes. Like C4.5, CART is a binary recursive partitioning procedure because parent nodes are always splitted into exactly two children nodes and each child node is treated as a parent in a recursive manner. As attribute selection metric, CART uses, in place of the Information Gain, the *Gini Index* defined as:

$$Gini(j) = 1 - \sum_{i=1}^{c} p_{i|j}^2 \quad (2)$$

where $c$ is the number of classes and $p_{i|j}$ the probability that an instance in node $j$ belongs to class $i$. In fact, if all instances of training sample $S$ belong to the same class, then $p_{i|j}$ =1 and the *Gini* index would be 0 corresponding to a node with no *impurity*. Like J48, the Simple Cart algorithm too can benefit from a pruning procedure as described in (Esposito, 1997).

### 4.1.3 Naïve Bayes

The Naïve Bayes classifier (Aggarwal, 2014) is based on the Bayes' rule and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, the Naïve Bayes classifier can in some cases achieve comparable performances to sophisticated classification methods based on neural networks. It assumes *class conditional independence* amongst attributes and decides according to the evidence provided by the training data; i.e., letting $x=(x_1,...,x_n)$ a vector of observed attributes and $C$ a class, the classifier decides in favour of $C_i$ class maximizing the a-posteriori probability:

$$P(C_i|x) \quad \frac{P(x|C_i)P(C_i)}{P(x)} \quad (3)$$

### 4.1.4 Random Forests

This algorithm, introduced by Breiman (Breiman, 2001), represents a combination of several individual trees designed so that, each time a split in a tree is considered, a random sample of $m$ predictors is chosen as split candidates from the full set of $p$ predictors. Tipically $m \approx \sqrt{p}$ as suggested in (James, 2014). The Random Forests derives from the Bagging and Boosting structures (Witten, 2005) where preliminary decisions are taken by different learners and thereafter combined. In Bagging, the final decision (if numeric) is taken on the basis of an average value, whereas, in Boosting, it is obtained as a weighted average of decisions of single learners working in a sequential manner. Random Forests just introduces a form of randomization in the Bagging structure resulting in a classifier output by running learner several times with different random number seeds and by combining the classifiers' predictions

through majority voting or averaging. In Random Forests there is no need for cross-validation nor for a separate test set to get an unbiased error estimate since the latter is produced internally while the algorithm runs.

Due to its characteristics, the Random Forests has been already applied successfully in anomaly and misuse detection systems [Zhang, 2008].

## 5 THE TESTBED

The following use case has been devised to assess the proposed system of WebRTC traffic detection. Two remote users start a real-time Voice/Video session by logging on *gruveo.com* portal, a web platform supporting the secure WebRTC standard so that privacy and confidentiality of the communication is preserved (similar results are expected from equivalent web platforms). An ad hoc gateway has been implemented which includes four modules (see Fig. 4):

- *Tshark* sensor to detect the DNS query to WebRTC service portal;
- *mysql* database to store information about users logged in;
- a conversion module to translate *pcap* files to CSV or ARFF formats;
- *Weka* module to perform the decision in the form of a boolean value (*WebRTC/Normal*);

The system is implemented as a Java-based finite state machine as shown in Fig. 4. The front-end consists of two main user interfaces. The first one, shown in Fig. 5, allows to select a classification algorithm (in fact, although we focused on J48, Simple CART, Naïve Bayes and Random Forests algorithms, the system easily adapts to additional algorithms) and to load the data gathered in a controlled environment for training purposes. The second interface, shown in Fig. 6, is aimed at continuously monitoring (with a 1/15 $sec^{-1}$ refresh rate) the IP addresses suspected to be involved in the WebRTC session. Once the training GUI loads the two *pcap* files making up the training dataset for the chosen algorithm, the system extracts from the acquired data the relevant statistics, i.e. the values of the attributes associated with each of the two classes, and stores them into an unique CSV file. After the training phase, the classifier starts to run in real-time with the objective to reveal if a targeted IP address is involved in a WebRTC call. In our experiment, a specific rule implemented through an embedded *TShark* routine works in background in order to intercept the initial logging phase of IP's making a DNS query to *gruveo.com* portal. Fig. 6 reproduces the display of the GUI while monitoring the encrypted WebRTC calls. In particular, the display shows the IP addresses logged on *gruveo.com* (in the left column), and the decision taken by the classifier acting on the traffic pertaining to that IP address (right column).

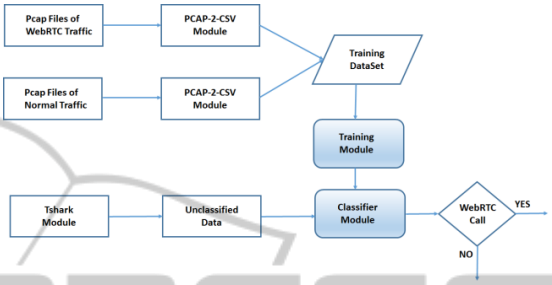


Figure 4: Schematic of the architecture.

For prototype purposes, we do not dwell on the specifications of the hardware hosting the proposed system; however, some performance analyses are available in (Witten, 2005).

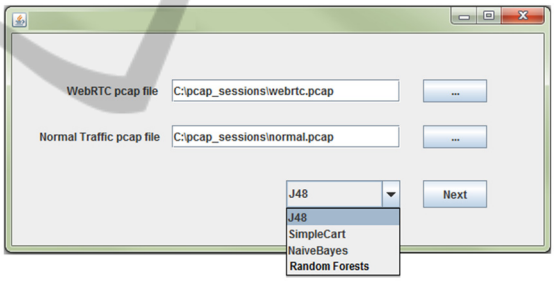


Figure 5: The implemented training GUI.

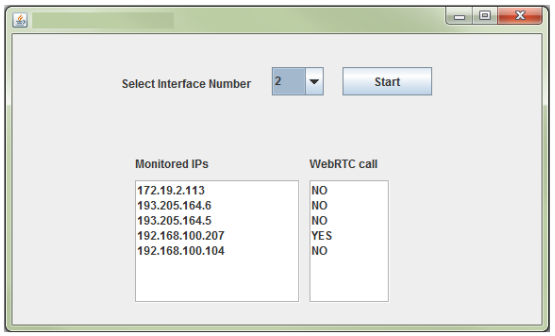


Figure 6: The implemented monitoring GUI.

## 6 EXPERIMENTAL RESULTS

The tests carried out in this research rely on the

logical architecture shown in Figure 4, conveniently deployed on a linux-based gateway. The sample space is made of records of attributes (as defined in 4.1) extracted from Internet traffic generated by a dedicated host. For our purposes, a training set worth of 1074 instances was first constructed and partitioned into the two classes:

1. *Normal*: A collection of 931 standard traffic (HTTP, FTP…) records
2. *WebRTC*: A collection of 143 records from encrypted WebRTC sessions

The different sizes of the two classes result in probabilities $p_{Normal} = 0.87$ and $p_{WebRTC} = 0.13$, values needed by the various algorithms for attribute selection and for decision. Once the system was trained through the *pcap* files as outlined in the previous section, the algorithms were tested for correctness in their respective running phases (for example, in a trial of 100 sessions, 50 *WebRTC* and 50 *Normal*, J48 achieved Detection Rate of 84% and a False Positive Rate of 21%) but the results are not significant at the moment due to the limited size of available test sets. More significant results have been obtained by exploiting the same training set through a 10-fold cross validation method offered natively by Weka (the folding factor follows a suggestion by (Aruna, 2001)). They are presented in Figures 7-10 for each algorithm in terms of confusion matrix and related statistics. Summarizing results are shown in Table 1 in terms of four statistics: *i)* the *True Positive Rate* indicating the proportion of positive cases that were correctly classified; *ii)* the *False Positive Rate* indicating the proportion of negative cases that were incorrectly classified as positive; *iii)* the *Precision* indicating the proportion of the predicted positive cases that ended up correct and, *iv)* the *F-Measure*, a measure of test's accuracy. An additional comparison of the aforementioned algorithms is presented in terms of ROC curves as shown in Figure 11 where the performance of each classifier is also summarized as the Area Under the Curve (AUC): the larger the AUC, the better the classifier.

Table 1: Comparison of metrics.

| | ***J48*** | ***Naïve Bayes*** | ***Simple Cart*** | ***Random Forests*** |
|---|---|---|---|---|
| ***TP Rate*** | 0,951 | 0,859 | 0,958 | 0,965 |
| ***FP Rate*** | 0,126 | 0,631 | 0,16 | 0,147 |
| ***Precision*** | 0,953 | 0,837 | 0,957 | 0,964 |
| ***F-Measure*** | 0,952 | 0,845 | 0,958 | 0,964 |

```
Correctly Classified Instances        1021               95.0652 %
Incorrectly Classified Instances        53                4.9348 %
Kappa statistic                          0.7941
Mean absolute error                      0.0686
Root mean squared error                  0.2061
Relative absolute error                 29.6327 %
Root relative squared error             60.6749 %
Total Number of Instances             1074

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.86      0.035      0.788      0.86      0.823      0.919    WebRTC
                 0.965     0.14       0.978      0.965     0.971      0.919    Normal
Avg.             0.951     0.126      0.953      0.951     0.952      0.919

=== Confusion Matrix ===

   a   b   <-- classified as
 123  20 |   a = WebRTC
  33 898 |   b = Normal
```

Figure 7: J48 algorithm results.

```
Correctly Classified Instances        1029               95.8101 %
Incorrectly Classified Instances        45                4.1899 %
Kappa statistic                          0.8147
Mean absolute error                      0.064
Root mean squared error                  0.1962
Relative absolute error                 27.6438 %
Root relative squared error             57.7451 %
Total Number of Instances             1074

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.818     0.02       0.86       0.818     0.839      0.907    WebRTC
                 0.98      0.182      0.972      0.98      0.976      0.907    Normal
Avg.             0.958     0.16       0.957      0.958     0.958      0.907

=== Confusion Matrix ===

   a   b   <-- classified as
 117  26 |   a = WebRTC
  19 912 |   b = Normal
```

Figure 8: Simple Cart algorithm results.

```
Correctly Classified Instances         923               85.9404 %
Incorrectly Classified Instances       151               14.0596 %
Kappa statistic                          0.2725
Mean absolute error                      0.1372
Root mean squared error                  0.3544
Relative absolute error                 59.3027 %
Root relative squared error            104.3152 %
Total Number of Instances             1074

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.28      0.052      0.455      0.28      0.346      0.894    WebRTC
                 0.948     0.72       0.896      0.948     0.921      0.888    Normal
Avg.             0.859     0.631      0.837      0.859     0.845      0.889

=== Confusion Matrix ===

   a   b   <-- classified as
  40 103 |   a = WebRTC
  48 883 |   b = Normal
```

Figure 9: Naïve Bayes algorithm results.

```
Correctly Classified Instances        1036               96.4618 %
Incorrectly Classified Instances        38                3.5382 %
Kappa statistic                          0.8421
Mean absolute error                      0.0519
Root mean squared error                  0.1717
Relative absolute error                 22.4485 %
Root relative squared error             50.5446 %
Coverage of cases (0.95 level)          99.2551 %
Mean rel. region size (0.95 level)      57.3091 %
Total Number of Instances             1074

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  ROC Area  Class
                 0,832     0,015     0,895      0,832     0,862      0,975    WebRTC
Weighted         0,985     0,168     0,974      0,985     0,980      0,975    Normal
  Avg.           0,965     0,147     0,964      0,965     0,964      0,975

=== Confusion Matrix ===

   a   b   <-- classified as
 119  24 |   a = WebRTC
  14 917 |   b = Normal
```

Figure 10: Random Forests algorithm results.

In particular we have:

$$Precision = \frac{t_p}{t_p + f_p} \quad (4)$$

where $t_p$ and $f_p$ are the true positives and negatives respectively and

$$F_{Measure} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (5)$$

where *Recall* is the proportion of actual positives correctly identified as such (equivalent to *True Positive Rate*). From the presented comparison it can be observed that Naïve Bayes is consistently outperformed by the other strategies, likely due the violation of the hypothesis of class conditional independence. The remaining three algorithms instead, achieve comparable performances, with J48 proving slightly superior in regard to *False Positive Rate* and Random Forests exhibiting better results in terms of *True Positive Rate*.

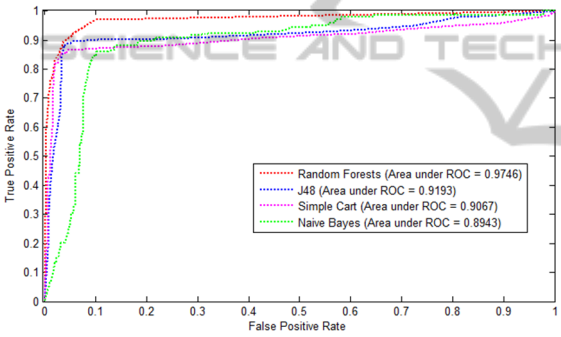


Figure 11: ROC curves and areas under ROCs (AUC).

## 7 CONCLUSIONS

WebRTC technology brings a noteworthy novelty in the field of web-based multimedia communications providing a peer-to-peer browser-based service for audio and video streaming without neglecting the security issues. To this aim, the embedded DTLS protocol, designed to prevent eavesdropping and information tampering, enables even the non-skilled user to setup an encrypted real-time multimedia session relying just on a common web browser. Due to such capability, the authorities in charge of lawful interception issues have to deal with new services that may elude traditional kinds of control. An audio/video session based on the encrypted WebRTC protocol in fact, is difficult to reveal because it can use dynamic ports allocation and does not include any characteristic pattern that allows a semantic-based recognition. In our paper we propose and assess an automatic decision system capable to reveal an encrypted WebRTC session with the support of Weka, a popular open source machine learning tool. The system includes a purposely developed Java engine to train the classifier based on a dataset containing traffic partitioned in the two classes envisioned: *WebRTC* and *Normal*. In our implementation we considered, as class attributes, some typical parameters derived during a real-time traffic processing phase such as: transport protocol, inter-arrival times, packet lengths and number of packets in forward and backward directions. Through a cross validation assessment scheme based on the training dataset, four among the most credited classification algorithms have been compared: J48, Simple Cart, Naïve Bayes and Random Forests. The experiment suggests that the Random Forests offer best results in terms of *True Positive Rate* whereas J48 performs better in terms of *False Positive Rate* detection. Future works will be aimed to compare a broader range of algorithms, including distributed ones, using large enough dataset to improve the significance of the results.

## REFERENCES

Aggarwal, C., 2014. *Data classification: algorithms and applications*. Taylor & Francis Press.

Aruna, S., Rajagopalan, S.P., Nandakishore, L.V., 2001. An Empirical comparison of Supervised learning algorithms in disease detection. In *IJITCS, Vol1, Nº4*.

Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K., 2004. IETF, RFC3711.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. *Classification and Regression Trees*. Taylor & Francis Press.

Breiman, L., 2001. Random Forests. In *Machine Learning Journal, Vol. 45, N°1, pp. 5-32*.

Dainotti, A., De Donato, W., Pescape, A., Rossi, P.S., 2008. Classification of Network Traffic via Packet-Level Hidden Markov Models. In *GLOBECOM'08, pp.1-5*. IEEE.

Di Mauro, M., Longo, M., 2014. Skype traffic detection: a decision theory based tool. In *ICCST'14, pp. 52-57*. IEEE.

Esposito, F., Malerba, D., Semeraro, G., 1997. A comparative analysis of methods for pruning decision trees. In *IEEE Transaction on Pattern analysis and machine intelligence, Vol. 19, No. 5*.

Freire, E. P., Ziviani, A., Salles R. M., 2008. Detecting Skype flows in Web traffic. In *NOMS'08, pp. 89-96*. IEEE.

James, G., Witten, D., Hastie, T., Tibshirani, R., 2014. *An introduction to statistical learning with applications in R*. Springer.

Kotsiantis, S. B., 2007. Supervised Machine Learning: A review of Classification Techniques. In *Emerging Artificial Intelligence Applications in Computer Engineering pp.3-24.* ACM.

Kuhn, M., Johnson, K., 2013. *Applied predictive modeling*. Springer.

Lennox, J., 2013. IETF, RFC6904.

Lin, P., Lei, Z., Chen, L., Yang, J., Liu, F., 2009. Decision tree network traffic classifier via adaptive hierarchical clustering for imperfect training dataset. In *WiCOM'09*, *pp. 1-6*. IEEE.

Loreto, S., Romano, S.P., 2014. *Real-time communication with WebRTC*. O'Reilly Media.

Mahy, R., Matthews, P., Rosenberg, J., 2010. IETF, RFC5766.

Meera Gandhi G., 2010. Machine Learning Approach for Attack Prediction and Classification using Supervised Learning Algorithms. In *IJCSNS pp. 247-250*.

Moore, A. W., Zuev, D., 2005. Internet traffic classification using Bayesian analysis techniques. In *SIGMETRICS'05, pp. 50-60.* ACM Press.

Nguyen, T. T., Armitage, G., 2008. A survey of techniques for Internet traffic classification using machine learning. In *Communications Surveys & Tutorials*, *Vol. 10, Issue: 4, pp. 56-76.* IEEE.

Quinlan, J. R., 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

Rahdari, F., Eftekhari, M., 2012. Using Bayesian classifiers for estimating quality of VoIP. *AISP12*, pp. 348-353.

Rescorla, E., Modadugu, N., 2006. Datagram Transport Layer Security. IETF, RFC4347.

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E., 2002. IETF, RFC 3261.

Rosenberg, J., Mahy, R., Matthews, P., Windg, D., 2008. IETF, RFC5389.

Ruggieri, S., 2002. Efficient C4.5. In *IEEE Transactions on Knowledge and Data Engineering. Vol. 14, Issue: 2*. IEEE.

Shi, Y., Hao, K., 2013. Design and realization of chatting tool based on web. In *CECNet'13, 3rd International Conference on Consumer Electronics, Communications and Networks, pp. 225-228.* IEEE.

Song, R., Liu, H., Xia, T., 2008. Study on signatures of P2P protocols based on regular expressions. In *ICALIP08*, *Audio, Language and Image Processing, pp. 863-867*. IEEE.

Witten, I., Frank, E., 2005. *Data Mining, practical Machine Learning Tools and Techniques*. Elsevier, 2nd edition.

Xusheng, Z., 2008. A P2P Traffic Classification Method Based on SVM. In *ISCSCT'08, International Symposium on Computer Science and Comutational Technology, vol. 2, pp.53-57*. IEEE.

Zeidan, A., Lehmann, A., Trick, U., 2014. WebRTC enabled multimedia conferencing and collaboration solution. In *WTC'14, World Telecommunications Congress, pp. 1-6*. IEEE.

Zhang, J., Zulkemine, M., Haque, A., 2008. Random-forests-based network intrusion detection systems. In *IEEE Transactions Systems, Man, Cybernetics C, Applications and Reviews, volume 38, no. 5, pp. 649–659*.IEEE.