# Markov Chain Fingerprinting to Classify Encrypted Traffic

Maciej Korczyński*¶ and Andrzej Duda¶
*EENR & DIMACS, Rutgers University, New Jersey, USA
¶Grenoble Institute of Technology, CNRS Grenoble Informatics Laboratory UMR 5217, France
Email: Maciej.Korczynski@rutgers.edu, Andrzej.Duda@imag.fr

***Abstract*—In this paper, we propose *stochastic fingerprints* for application traffic flows conveyed in Secure Socket Layer/Transport Layer Security (SSL/TLS) sessions. The fingerprints are based on first-order homogeneous Markov chains for which we identify the parameters from observed training application traces. As the fingerprint parameters of chosen applications considerably differ, the method results in a very good accuracy of application discrimination and provides a possibility of detecting abnormal SSL/TLS sessions. Our analysis of the results reveals that obtaining application discrimination mainly comes from incorrect implementation practice, the misuse of the SSL/TLS protocol, various server configurations, and the application nature.**

## I. Introduction

The importance of appropriate traffic classification methods continues to grow. They are essential for effective network planning, policy-based traffic management, application prioritization, and security control. However, traditional port-based [1] and payload-based [2], [3] classification methods become less effective, because new applications can hide their nature by dynamically assigning ports, by using tunneling, or by applying proprietary payload encryption methods. This situation has led to the development of new identification methods based on flow features [4], [5] and host behavior [6], [7]. They are useful for classification of application layer protocols (e.g. HTTP, DNS, BitTorrent, etc.) or traffic categories (e.g. P2P content sharing, games, multimedia, WWW, etc.). However, they are less effective for reliable identification of application flows on top of a given protocol. Moreover, apart from some notable exceptions, these methods are not appropriate for classifying traffic where only one direction is observed due to routing asymmetries [8].

The past research on traffic analysis and classification showed that once we are able to generate a unique signature based on the packet or message payload (e.g. HTTP request headers), we can classify applications with high accuracy [3], [8]. Unfortunately, such approaches fail in case of encrypted traffic [9]. In this work, we propose a payload-based method to identify application flows encrypted with the Secure Socket Layer/Transport Layer Security (SSL/TLS) protocol, which is a fundamental cryptographic protocol suite supporting secure communication over the Internet [10].

Our approach consists of taking advantage of the information embedded in the SSL/TLS header to create statistical *fingerprints* of sessions to classify application traffic. We call a fingerprint any distinctive feature allowing identification of a given traffic class. In this work, a fingerprint corresponds to a first-order homogeneous Markov chain reflecting the dynamics of an SSL/TLS session. The Markov chain states model a sequence of SSL/TLS message types appearing in a single direction flow of a given application from a server to a client.

We have studied the Markov chain fingerprints for twelve representative applications that make use of SSL/TLS: PayPal (an electronic service allowing online payments and money transfers), Twitter (an online social networking and microblogging service), Dropbox (a file hosting service), Gadu-Gadu (a popular Polish instant messenger), Mozilla (a part of Mozilla add-ons service responsible for verification of the software version), MBank and PKO (two popular European online banking services), Dziekanat (student online service), Poczta (student online mail service), Amazon S3 (a Simple Storage Service) and EC2 (an Elastic Compute Cloud), and Skype (a VoIP service). The resulting models exhibit a specific structure allowing to classify encrypted application flows by comparing its message sequences with fingerprints. They can also serve to reveal intrusions trying to exploit the SSL/TLS protocol by establishing abnormal communications with a server.

## II. SSL/TLS Overview

Secure Sockets Layer (SSL) [11] and its successor Transport Layer Security (TLS) [10] are cryptographic protocols that provide secure communication between two parties over the Internet by encapsulating and encrypting application layer data. Many WWW portals and servers, especially those providing commercial services, use SSL/TLS for guaranteeing security of all operations.

Figure 1 illustrates the structure of SSL/TLS and its components:

- *Record Protocol*: compresses and encrypts upper-layer data using the security parameters configured by the Handshake Protocol.
- *Application Data Protocol*: provides application layer data to the Record Protocol.
- *Handshake Protocol*: negotiates parameters of an SSL/TLS session. Two communicating parties agree on the protocol version to use, they optionally authenticate each other, exchange information on the session ID, select

978-1-4799-3360-0/14/$31.00 

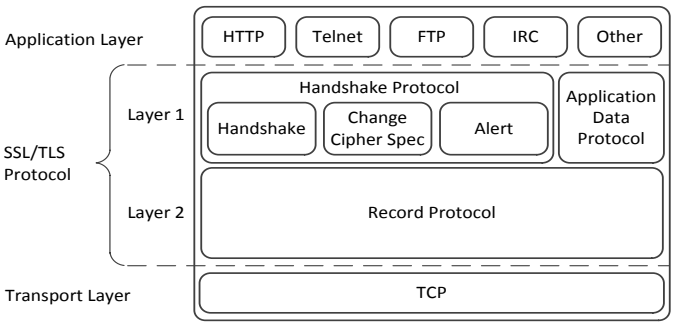


Figure 1. SSL/TLS protocol structure

cryptographic and compression algorithms, as well as the shared secret used to generate keys.

- *Change Cipher Spec Protocol*: signals modifications to encryption strategies. The protocol consists of a single message sent by either the client or the server to inform the other party that successive records will use the newly negotiated cryptographic algorithm and keys.
- *Alert Protocol*: reports an error condition or a change in status of the session.

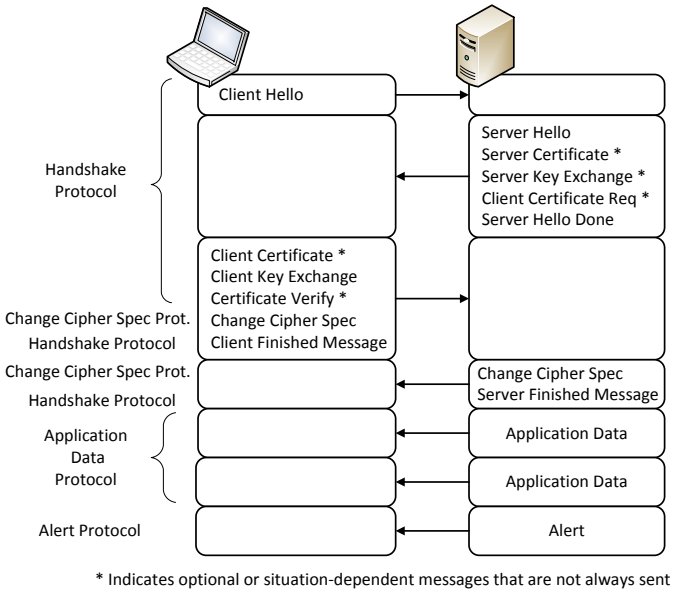


Figure 2. Message exchange during an SSL/TLS session with a full handshake

Figure 2 presents an example message exchange between a client and a server during the SSL/TLS session with a full handshake.

The initial message exchange of `Client Hello` and `Server Hello` establishes the attributes: Protocol Version, Session ID, Cipher Suite, and Compression Method. The key exchange uses up to four messages: server `Certificate`, `Server Key Exchange`, client `Certificate`, and `Client Key Exchange`. Then, the client sends `Change Cipher Spec` and the next `Finished` message is encrypted with the new algorithms and keys. In response, the server sends its own `Change Cipher Spec` message and the `Finished` message under the new cipher specification, which completes the SSL/TLS handshake and the two parties can exchange application layer data. The server terminates the session with an `Alert` message.

The exchange is an example—a session can be shortened by resuming previous sessions using the Session ID or it can be significantly modified depending on server configuration and application requirements. Note that during the SSL/TLS handshake much information is sent as plaintext. However, after the `Server Hello Done` or `Change Cipher Spec` protocol message, only the protocol type, the length of a record, and the SSL/TLS version are not encrypted.

We use below the following compact notation of message types: the decimal protocol types and, if not encrypted, the corresponding message types present in the SSL/TLS headers (cf. Figure 3). For instance, we represent the `Application Data Protocol` as 23: and the `Handshake Client Hello` message as 22:1.

In our study, we consider only server-side message types of an SSL/TLS session. Depending on client configurations

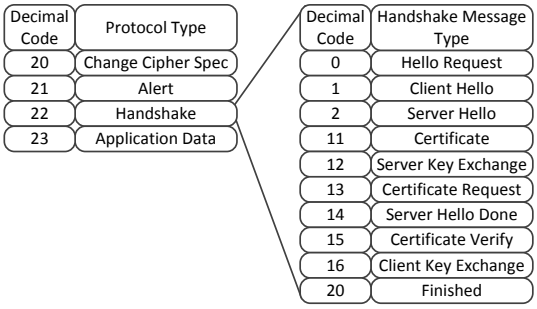


Figure 3. SSL/TLS protocol types and their corresponding decimal codes

(e.g. SSL/TLS protocol settings of Web browsers), we expect slightly different characteristics for the client side, whereas the service-side model should be representative of all networks. Moreover, the separation of client and server-side models helps tackling the problem of asymmetric routing when we can only observe traffic in one direction.

## III. Markov Chain Fingerprints

In this section, we propose an approach based on Markov chains to model possible sequences of message types observed in single-directional SSL/TLS sessions. We have chosen a first-order homogeneous Markov chain model due to its simplicity.

We consider discrete-time random variable $X_t$ for any $t = t_0, t_1, ..., t_n \in T$. It takes values $i_t \in \{1, ..., s\}$, where $i_t$ is either an SSL/TLS message type (e.g. 22:2) or a sequence of the SSL/TLS message types transmitted in a single TCP segment (e.g. 22:11,22:14).

We assume that $X_t$ is a first-order Markov chain [12]:

$$P(X_t = i_t | X_{t-1} = i_{t-1}, X_{t-2} = i_{t-2}, \ldots, X_1 = i_1) = P(X_t = i_t | X_{t-1} = i_{t-1}). \quad (1)$$

We further assume that the Markov chain is homogeneous, i.e. a state transition from time $t-1$ to time $t$ is time-invariant:

$$P(X_t = i_t | X_{t-1} = i_{t-1}) = P(X_t = j | X_{t-1} = i) = p_{i-j}, \quad (2)$$

with the transition matrix [12]:

$$P = \begin{bmatrix} p_{1-1} & p_{1-2} & \cdots & p_{1-s} \\ p_{2-1} & p_{2-2} & \cdots & p_{2-s} \\ \vdots & \vdots & \ddots & \vdots \\ p_{s-1} & p_{s-2} & \cdots & p_{s-s} \end{bmatrix}, \quad (3)$$

where: $\sum_{j=1}^{s} p_{i-j} = 1$. We denote by:

$$Q = [q_1, q_2, \ldots, q_s], \quad (4)$$

the ENter Probability Distribution (ENPD), where $q_i = P(X_t = i)$ at time $t_0$, and we define:

$$W = [w_1, w_2, \ldots, w_s], \quad (5)$$

as the EXit Probability Distribution (EXPD), where $w_i$ represents the probability that the session finishes when it is in state $i$ at time $t_n$. Note that both probability distributions are independent of the Markov chain—they provide the probabilities to enter and quit the Markov chain. In traditional Markov chain models, there is an initial state and one or several absorbing states. In our case, ENPD defines the probability to enter one of the state of the Markov chain and EXPD gives the probability of quitting the Markov chain from any of its states. Based on these definitions, the probability that a sequence of states $X_1, \ldots, X_T$ representing a single SSL/TLS session occurs is as follows:

$$P(\{X_1, \ldots, X_T\}) = q_{i_1} \times \prod_{t=2}^{T} p_{i_{t-1}-i_t} \times w_{i_T}. \quad (6)$$

The resulting probability indicates how a given SSL/TLS sequence of message types during a session is close to a model of an application flow: a larger value means that the SSL/TLS session is closer to the model.

To illustrate the process of the fingerprint creation, consider the following examples of the message sequences observed during SSL/TLS sessions in a training dataset composed of only three server-side SSL/TLS flows of the PayPal application traffic:

```
22:2-22:11,22:14-20:,22:-23:
22:2,20:,22:-23:
22:2-22:11,22:14-20:,22:-23:-23:-21:
```

There are 6 different Markov states in the example. The transition probability between states is derived from frequencies observed in the sequences, e.g. $P_{22:2-22:11,22:14} = 1$, while $P_{23:-23:} = 0.5$. The ENPD vector is composed of two non-zero elements, namely $P_{22:2} = 0.67$ and $P_{22:2,20:,22:} = 0.33$, whereas the EXPD vector also contains two non-zero elements $P_{23:} = 0.67$ and $P_{21:} = 0.33$. The probabilities are the parameters of the Markov chain fingerprint for the PayPal traffic. Based on the model, we can find the probability that an observed SSL/TLS session conveys the PayPal application traffic (cf. Eq. 6). The probability that the following sequence of SSL/TSL message types:

```
22:2,20:,22:-23:-23:-23:
```

is a PayPal flow is equal to: $P(\{X_1, \ldots, X_4\}) = 0.055$. In comparison, the probability computed from the Twitter model (cf. Figure 5) is equal to 0.003%, whereas the probability computed from the Skype fingerprint (cf. Figure 9) is equal to 0.

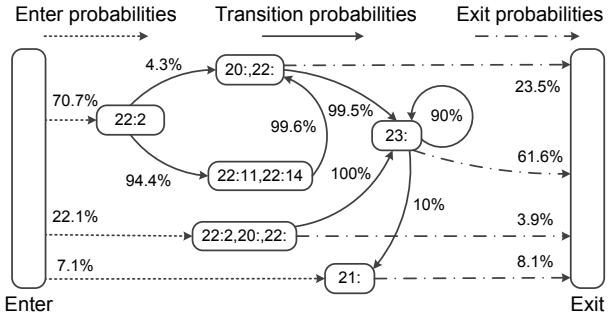


Figure 4. Parameters of the fingerprint for PayPal

### A. Examples of Fingerprints

Figures 4-9 illustrate the fingerprints derived for chosen application traffic—they represent ENPD, the transition probability matrix, and EXPD. The diagrams are simplified for clarity by including only the states with meaningful probabilities (full models usually contain a large number of states). For this reason, in some cases, ENPD and EXPD do not sum to 100%. Due to the space limitation, we only provide six models out of twelve analyzed applications: PayPal, Twitter, Dropbox, Mozilla, Gadu-Gadu, and Skype. The model parameters are derived from a representative dataset (cf. Section IV-A for the description of the `Campus2` dataset). For Skype, the measured data comes from the traffic dataset recorded for Skype service flow classification [13].

*1) PayPal:* Figure 4 for PayPal shows that 92.8% of all sessions start with the `Server Hello` message, whereas 7.1% are `Alert` messages indicating handshake failure and closing the session even before the authentication process. Moreover, in case of successfully established sessions, the server always sends `Change Cipher Spec` indicating modifications in ciphering strategies. In 66.74% of sessions, the client authenticates the server, whereas in the remaining cases, the session is resumed using the Session ID (the handshake procedure is shortened to the `Server Hello` message). Finally, from the Exit Probability Distribution, we can conclude that in most cases, successful sessions do not end up with the `Alert` message coming from the server.

*2) Twitter:* Figure 5 indicates that 55% of SSL/TLS sessions are resumed from previously negotiated ones. Contrary to PayPal, almost 70% of remaining sessions do not change ciphering strategy after the server authentication procedure (no `Change Cipher Spec` message). Moreover, sessions tend

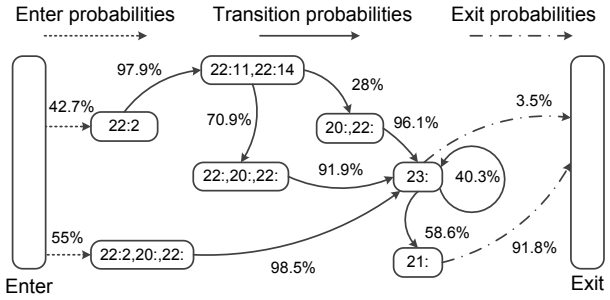


Figure 5. Parameters of the fingerprint for Twitter

to be rather short and are composed of `Application Data` message followed by the `Alert` message (with probability 58.6%) terminating the session.

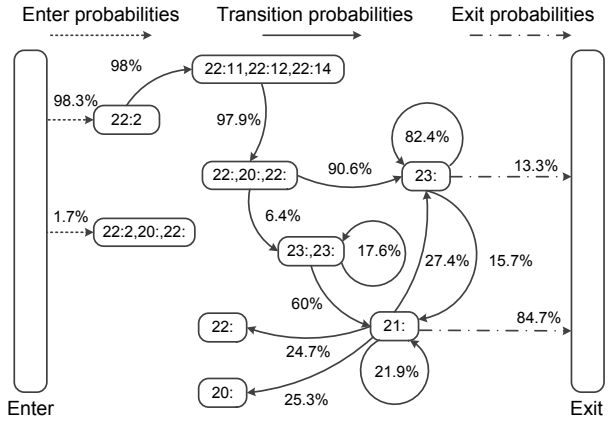


Figure 6. Parameters of the fingerprint for Dropbox

*3) Dropbox:* Contrary to previously presented PayPal and Twitter, the great majority of initial sessions (98.3%) never resume previous SSL/TLS sessions as shown in Figure 6. Furthermore, we can observe the `Server Key Exchange` message that contains additional cryptographic information to the `Certificate` message allowing the client to communicate the premaster secret. Sessions are composed of multiple `Application Data` messages, which reflects the specific nature of the application. Again, the majority of sessions (84.7%) are terminated by sending the `Alert` message. Despite the fact that sessions are highly consistent and message sequences often repeat, we can observe quite a few unusual states signaled by the Alert Protocol.

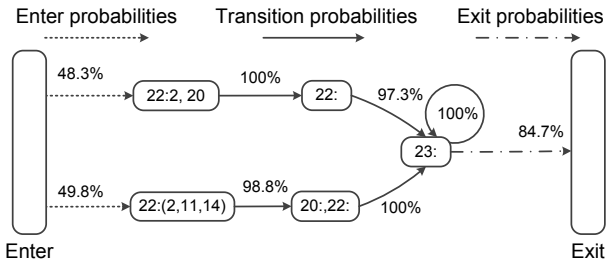


Figure 7. Parameters of the fingerprint for Mozilla

*4) Mozilla:* In Figure 7 for Mozilla, we can observe a rare initial state—so-called a multiple handshake message in which we have identified three messages in a single TCP segment with SSL/TLS handshake, namely `Server Hello`, `Certificate`, and `Server Hello Done`, depicted as 22:(2,11,14). Also, the number of significant states is limited to five.

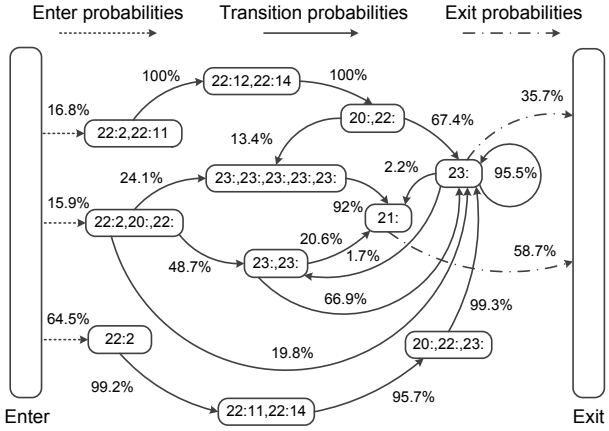


Figure 8. Parameters of the fingerprint for Gadu-Gadu

*5) Gadu-Gadu:* As we can observe in Figure 8, Gadu-Gadu presents three possibilities to establish a session. The primary one (64.5%) consist of a typical `Server Hello` message followed by the `Certificate` and `Server Hello Done` messages. In 95.7% cases, the `Change Cipher Spec` message comes after. The second SSL/TLS handshake procedure additionally includes the `Server Key Exchange` message. Finally, 15.9% sessions are being resumed. The figure suggests that on the average, Gadu-Gadu sessions consist of a significantly larger number of messages in comparison to PayPal, Mozilla, or Twitter. Moreover, we observe individual segments composed of multiple `Application Data` messages, which presumably implies that application layer messages are relatively short—this feature stands out from the previous cases.

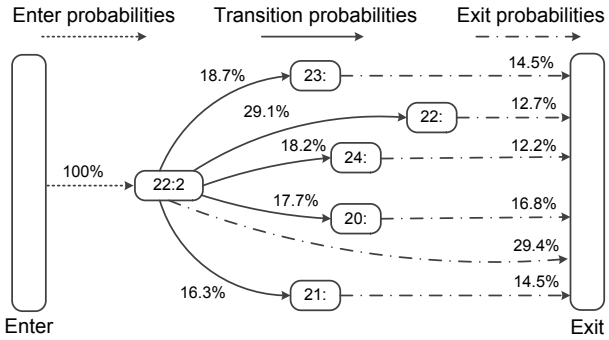


Figure 9. Parameters of the fingerprint for Skype

*6) Skype:* Finally, we present the example of the Markov chain analysis of the Skype traffic tunneled through SSL/TLS (cf. Figure 9). Skype traffic represents a special case—the

Markov state space contains only six states. In addition to four standard SSL/TLS protocol type messages, we do observe a unique state interpreted by Wireshark as a Heartbeat Protocol message defined in RFC 6520 [14], depicted as 24:. Briefly, the Heartbeat Extension provides a new protocol for TLS allowing the use of keep-alive functionality without performing a session renegotiation. In this case however, all five protocols act as application data protocols and directly provide application layer data to the Record Protocol. After the transition from the initial state to one of five remaining states with probability 70.6%, the transition probabilities between each of them are very similar ranging from 18.9% to 21.2% (not depicted in the figure for the sake of clarity).

Actually, Skype is a proprietary piece of software that uses its own internal encryption mechanisms and a complex connection protocol designed for bypassing firewalls and establishing communication regardless of network policies [13]. Skype randomly selects ports and can switch to port 443 if it fails to establish a connection on chosen ports. Such technique is sufficient to bypass network-layer firewalls, however, it results in creating a particular SSL/TLS session and a salient Markov fingerprint.

### B. Cross-validation

To validate the constructed models, we apply a 4-fold cross-validation based on four heterogeneous datasets described in Section IV-A. More precisely, we create Markov chains, compute the ENPD and EXPD vectors for application traffic based on one dataset (training set) and validate the analysis on the remaining three datasets (testing sets).

We perform validation as follows. First, we pre-process the testing set to extract application flows and then the classifier applies a decision process based on the Maximum Likelihood criterion [15]. Validation based on models obtained in the training process corresponds to a multi-hypothesis decision problem. More specifically, we consider up to twelve hypothesis $H_i, i = 1, \ldots, 12$ corresponding to each of considered applications. We apply a classical approach based on Maximum Likelihood criterion—we select the hypothesis under which the data sequence $Y$ is most likely:

$$H = \arg\max_{H_i} \log L(\{Y_1, \ldots, Y_T\}|H_i), \tag{7}$$

where $L(\{Y_1, \ldots, Y_T\})$ is the likelihood of the input data sequence under each hypothesis: $L(\{Y_1, \ldots, Y_T\}) \equiv P(\{X_1, \ldots, X_T\})$, the probability of a message sequence computed over each fingerprint (cf. Eq. 6).

## IV. Classification Results

In this section, we present the results of cross-validation of the fingerprints based on four trace datasets.

### A. Datasets

To build and validate fingerprints, we have used four recent heterogeneous datasets gathered on edge routers located in a European country. The `Campus1` and `Campus2` datasets come from two links connecting a large campus network to the Internet. `Campus1` dataset contains a one day long trace starting from March 1, 2012, whereas the 24 hours long `Campus2` dataset was obtained starting from March 26, 2012. The datasets labeled `Campus3` and `Campus4` consist of data observed at a different part of the campus network than the previous ones and were collected starting from July, 17, and July 21, 2013, lasting one day and 42 hours, respectively. All datasets contain only SSL/TLS encrypted traffic generated by standard services such as Web, chat, mail, VoIP, file transfer, or streaming applications. Moreover, we often refer to Skype as an example of traffic tunneled through SSL/TLS. The evaluation runs on two sets of packet traces generated in the experiments of classifying Skype service flows [13]. We merged two Skype datasets with `Campus1` and `Campus2` packet traces. For privacy reason, we analyze a TCP payload and export only information from SSL/TLS headers, while the actual payload is discarded.

### B. Ground Truth

To establish the ground truth, we have developed a *Domain Name System Classifier (DNSC)* to extract SSL/TLS application flows according to their domain names. More specifically, DNSC matches hostnames to an array of signature strings such as *twitter, r-*twttr* in case of Twitter. Table I summarizes domains used by DNSC for identifying different SSL/TLS applications. We have confirmed the accuracy of the method by manual payload inspection. Nevertheless, we might not cover all instances of signatures for a particular application. Another constraint of the approach is that we cannot obtain the instances of applications, if we are not able to resolve IP addresses into the corresponding domain names. For example, we cannot extract Skype flows using DNSC because the application relies on a Peer-to-Peer (P2P) infrastructure and the traffic is relayed by ordinary hosts.

Table I
DOMAIN NAMES USED BY DNSC. IRRELEVANT STRINGS ARE REPLACED BY AN ASTERISK.

| Application | Strings |
|---|---|
| PayPal | `*active*paypal*` |
| Twitter | `*twitter.com, r-*twttr` |
| Dropbox | `*dropbox*` |
| Gadu-Gadu | `ip*gadu-gadu.pl` |
| Mozilla | `*versioncheck*mozilla*` |
| MBank | `www.mbank*` |
| PKO | `ipko*` |
| Dziekanat | `dziekanat.agh.edu.pl` |
| Poczta | `poczta.agh.edu.pl` |
| Amazon S3 | `s3*amazon*` |
| Amazon EC2 | `ec2*amazon*` |

### C. Application Selection

In our experimental evaluation, to overcome the limitations of the DNSC classifier, we have selected the applications for which the IP address resolution was possible and corresponding strings are straightforward and unambiguous.

Table II presents the number of flows derived from the training datasets for the purpose of cross-validation of fingerprints. To estimate the minimal number of flows required

Table II
NUMBER OF FLOWS CORRESPONDING TO EACH APPLICATION IN FOUR DATASETS.

| Application | Campus1 | Campus2 | Campus3 | Campus4 |
|---|---|---|---|---|
| PayPal | 546 | 421 | – | – |
| Twitter | 1257 | 1500 | 8848 | 10308 |
| Dropbox | 1160 | 3134 | 4714 | 5253 |
| Gadu-Gadu | 659 | 807 | 1318 | 1779 |
| Mozilla | 1017 | 1076 | 2431 | 1567 |
| MBank | 644 | 94 | 675 | 459 |
| PKO | 354 | 420 | 1574 | 1137 |
| Dziekanat | 1162 | 1706 | 2655 | 609 |
| Poczta | 680 | 944 | 944 | 4420 |
| Amazon S3 | 238 | 321 | 1587 | 1310 |
| Amazon EC2 | 109 | 314 | 5798 | 610 |
| Skype | 210 | 207 | – | – |

Table III
CLASSIFICATION RESULTS FOR THE FINGERPRINTS. TRAINING DATASET: CAMPUS1, VALIDATION DATASETS: CAMPUS2, CAMPUS3, CAMPUS4

| | Campus2 | | Campus3 | | Campus4 | |
|---|---|---|---|---|---|---|
| Application | TPR | FPR | TPR | FPR | TPR | FPR |
| PayPal | 0.76 | 0.007 | – | – | – | – |
| Twitter | 0.932 | 0.013 | 0.768 | 0.029 | 0.791 | 0.023 |
| Dropbox | 0.922 | 0.001 | 0.971 | 0.007 | 0.957 | 0.009 |
| Gadu-Gadu | 0.865 | 0.001 | 0.535 | 0.053 | 0.59 | 0.063 |
| Mozilla | 0.998 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| MBank | 0.67 | 0.03 | 0.008 | 0.025 | 0.0 | 0.01 |
| PKO | 0.957 | 0.016 | 0.916 | 0.164 | 0.92 | 0.12 |
| Dziekanat | 0.807 | 0.005 | 0.83 | 0.0 | 0.805 | 0.001 |
| Poczta | 0.976 | 0.025 | 0.97 | 0.008 | 0.97 | 0.02 |
| Skype | 0.986 | 0.002 | – | – | – | – |

to create a reliable application fingerprint, we have applied the following procedure: we start with a model based on a randomly chosen flow. We build the state space and the transition matrix corresponding to a first-order homogeneous Markov chain. Therefore, we enrich the model by randomly including flows one by one and we observe the stability of the model. When the number of states, transitions and transition probabilities do not significantly change when enriching the model, the fingerprint can be included in the validation process. Depending on the application, the minimal number of the required flows may vary. However, even if the number of flows for some application is not sufficient to create a model, e.g. Amazon S3 or EC2 in Campus1 or Campus2, they can still serve to validate fingerprints built upon other datasets.

Table IV
CLASSIFICATION RESULTS FOR THE FINGERPRINTS. TRAINING DATASET: CAMPUS2, VALIDATION DATASETS: CAMPUS1, CAMPUS3, CAMPUS4

| | Campus1 | | Campus3 | | Campus4 | |
|---|---|---|---|---|---|---|
| Application | TPR | FPR | TPR | FPR | TPR | FPR |
| PayPal | 0.749 | 0.007 | – | – | – | – |
| Twitter | 0.847 | 0.003 | 0.438 | 0.012 | 0.528 | 0.007 |
| Dropbox | 0.984 | 0.009 | 0.987 | 0.056 | 0.985 | 0.06 |
| Gadu-Gadu | 0.975 | 0.015 | 0.521 | 0.146 | 0.569 | 0.153 |
| Mozilla | 0.988 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| PKO | 0.901 | 0.027 | 0.889 | 0.141 | 0.898 | 0.098 |
| Dziekanat | 0.997 | 0.002 | 1.0 | 0.005 | 0.995 | 0.006 |
| Poczta | 0.961 | 0.003 | 0.968 | 0.009 | 0.969 | 0.019 |
| Skype | 0.986 | 0.001 | – | – | – | – |

### D. Criteria of Cross-validation

We assume that the classification based on the DNSC reference classifier provides a reliable benchmark and we validate SSL/TLS models with respect to its classification decisions. We consider two meaningful metrics to assess the performance of the classification method: the *true positive (TP)* and *false positive (FP)* rates (denoted TPR and FPR, respectively). True Positive occurs when the validation result is consistent with the classification decision taken by DNSC and the application session is correctly classified as a given application, e.g. a PayPal session is accurately recognized as PayPal. Conversely, False Positive occurs when the validation result is inconsistent with the decision taken by the reference classifier and a session is incorrectly classified, e.g. a Twitter session is falsely recognized as PayPal.

### E. Cross-validation Results

Section III-A has shown that we can observe a great variety of SSL/TLS message exchanges. The parameters of the derived fingerprints differ considerably, which is the basis for accurate application discrimination. In this section, we report on the 4-fold cross-validation results of the application models in which Campus1−4 datasets served for training and validation alternately (cf. Table III-VI).

Let us take the example of Amazon S3, for which we have observed that the TP rate exceeds 97% regardless of the classification datasets used for training and validation (cf. Table V and VI). By manual inspection, we have found a multiple handshake message state observed previously in the Mozilla models built from the packet traces collected in March of 2012 (cf. Figure 7). This is also the reason why we experience a higher FP rate when validating Amazon S3 models on the Campus1 and Campus2 datasets.

Table V
CLASSIFICATION RESULTS FOR THE FINGERPRINTS. TRAINING DATASET: CAMPUS3, VALIDATION DATASETS: CAMPUS1, CAMPUS2, CAMPUS4

| | Campus1 | | Campus2 | | Campus4 | |
|---|---|---|---|---|---|---|
| Application | TPR | FPR | TPR | FPR | TPR | FPR |
| Twitter | 0.932 | 0.007 | 0.908 | 0.04 | 0.907 | 0.018 |
| Dropbox | 0.692 | 0.01 | 0.704 | 0.005 | 0.922 | 0.01 |
| Gadu-Gadu | 0.97 | 0.004 | 0.916 | 0.004 | 0.781 | 0007 |
| Mozilla | 0.001 | 0.028 | 0.0 | 0.023 | 0.405 | 0.041 |
| MBank | 0.02 | 0.006 | 0.0 | 0.007 | 0.817 | 0.018 |
| PKO | 0.597 | 0.005 | 0.537 | 0.004 | 0.595 | 0.035 |
| Dziekanat | 0.966 | 0.093 | 0.933 | 0.012 | 0.988 | 0.0 |
| Poczta | 0.942 | 0.002 | 0.967 | 0.004 | 0.97 | 0.0 |
| Amazon S3 | 0.978 | 0.146 | 0.991 | 0.111 | 0.996 | 0.0 |
| Amazon EC2 | 0.02 | 0.007 | 0.035 | 0.084 | 0.579 | 0.013 |

To better understand the results for Dropbox, let us consider its architecture and operation. The control and data storage servers are two major components of its architecture [16]. While the former is controlled by Dropbox Inc., the latter is managed by the Amazon S3 and EC2 servers. As we could expect, in some cases, Dropbox flows are incorrectly classified as Amazon EC2 resulting in a lower TPR (cf. Table V and VI). However, the classification based on models built upon the most recent flows coming from the Campus3 and Campus4

Table VI
CLASSIFICATION RESULTS FOR THE FINGERPRINTS. TRAINING DATASET: CAMPUS4, VALIDATION DATASETS: CAMPUS1, CAMPUS2, CAMPUS3

| | Campus1 | | Campus2 | | Campus3 | |
|---|---|---|---|---|---|---|
| Application | TPR | FPR | TPR | FPR | TPR | FPR |
| Twitter | 0.936 | 0.01 | 0.911 | 0.041 | 0.887 | 0.026 |
| Dropbox | 0.672 | 0.005 | 0.7 | 0.005 | 0.919 | 0.06 |
| Gadu-Gadu | 0.975 | 0.011 | 0.929 | 0.01 | 0.684 | 0.013 |
| Mozilla | 0.001 | 0.029 | 0.0 | 0.023 | 0.29 | 0.035 |
| MBank | 0.0 | 0.013 | 0.0 | 0.01 | 0.903 | 0.037 |
| PKO | 0.521 | 0.005 | 0.489 | 0.003 | 0.575 | 0.032 |
| Dziekanat | 0.959 | 0.092 | 0.929 | 0.012 | 0.994 | 0.0 |
| Poczta | 0.924 | 0.002 | 0.935 | 0.003 | 0.97 | 0.0 |
| Amazon S3 | 0.982 | 0.142 | 0.99 | 0.112 | 0.994 | 0.0 |
| Amazon EC2 | 0.146 | 0.044 | 0.001 | 0.079 | 0.598 | 0.01 |

datasets lead to TPR higher than 90%. The accurate fingerprint comes from the nature of Dropbox sessions: they exchanged a lot of data after the handshake process, which results in long SSL/TLS sessions composed of multiple application data protocol messages.

We have found a reliable fingerprint for the Poczta application for all four datasets. The most commonly observed state in SSL/TLS exchanges is composed of four application data protocol messages merged together: 23:23:23:23:.

The Skype traffic tunneled over the SSL/TLS protocol results in a unique fingerprint. The number of transitions depends on the service such as voice calls, chat, skypeOut or file sharing, and the amount of data to be sent. Every few to tens of seconds, Skype encapsulates a huge portion of data (in comparison to typical values) ranging from 3KB up to 65KB in one of its 6 SSL/TLS protocol types. Such an SSL/TLS segment is further divided into multiple TCP segments and sent across the network. This behavior is consistent with the real-time nature of Skype—creating multiple SSL/TLS messages could potentially increase the processing time and influence the quality of experience.

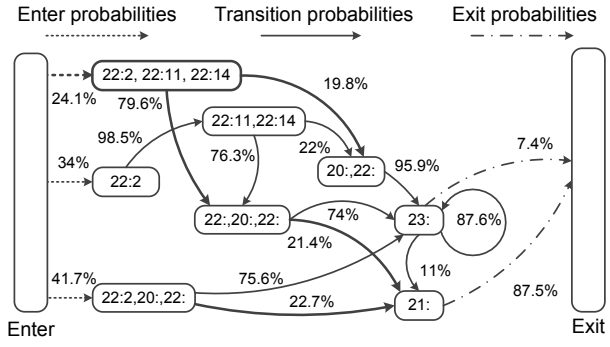


Figure 10. Parameters of the fingerprint for Twitter

Although, the proposed methodology results in very reliable application fingerprints, it may fail in some cases. For instance, we have identified a specific Markov chain instance in the fingerprint for Amazon EC2. It is composed of 5 states and more than 53% of all of its SSL/TLS sessions generate the same transition chain. The drawback is that the probabilities of all other chain instances of Amazon EC2 are very low, which results in a false negative rate not lower than 40% (cf. Table V and VI).

### F. Time Evolution of Markov Fingerprints

We have run our tests twice during the last one year and a half to study the evolution of the SSL/TLS fingerprints. By analyzing Markov chains, we can also extract a variety of meaningful information and possibly evaluate the changes in cryptographic practices of application servers. For example, we can clearly observe that the classification based on the fingerprints generated for Twitter and Gadu-Gadu in March 2012 (datasets `Campus1` and `Campus2`) results in a worse TPR compared to the validation on two recent `Campus3` and `Campus4` datasets. Moreover, the classification based on the Mozilla and MBank fingerprints failed because of implementation changes between the two observation periods.

Let us focus on the Twitter application. Figure 10 presents the fingerprint based on one of two most recent datasets, namely `Campus3`. To emphasize the fingerprint differences between the older model (cf. Figure 5) and the recent one, we have thickened the new states and transitions. When comparing with the fingerprint based on traffic collected one year earlier, we can notice only a change in the SSL/TLS segmentation. More specifically, we can observe a new state in the ENDP vector created by merging two neighboring states from the older model. In this case, enriching the model with recent application flows can significantly improve classification performance.

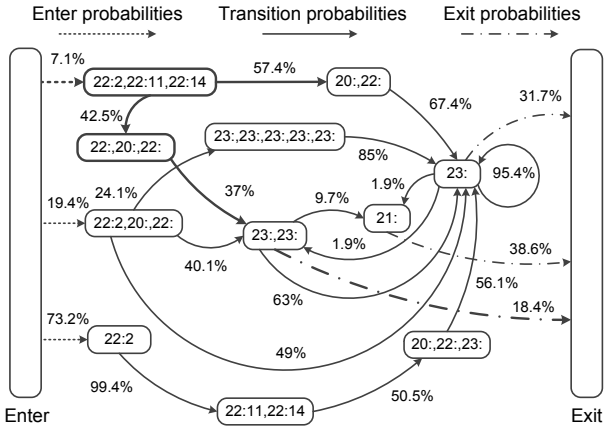


Figure 11. Parameters of the fingerprint for Gadu-Gadu

Figure 11 presents a recent fingerprint for Gadu-Gadu. When comparing with the older model in Figure 8, the primary transition composed of the `Server Hello` and `Certificate` messages followed by a state that consists of the `Server Key Exchange` and `Server Hello Done` messages is replaced by one initial state composed of the `Server Hello`, `Certificate`, and `Server Hello Done` messages. In other words, in the recent Gadu-Gadu fingerprints, we have not observed either `Server Key Exchange` or the associated Diffie-Hellman (DHE) key exchange algorithm. It means that either the application has

changed the key exchange methods and does not support the DHE algorithm anymore or, which is less likely, all observed clients have used the RSA, DH_DSS, or DH_RSA key exchange methods in which the `Server Key Exchange` message is not allowed [10].

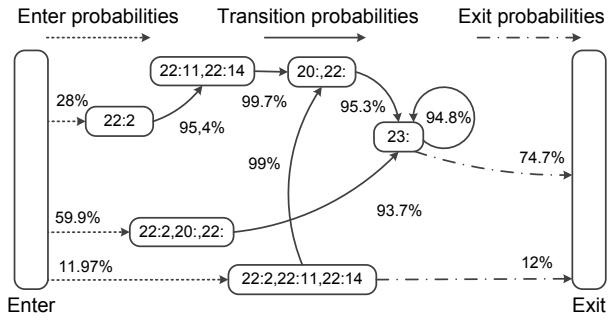


Figure 12. Parameters of the fingerprint for Mozilla

Finally, as we have expected by analyzing the cross-validation results, a simple "service version check" maintained by Mozilla has changed the SSL/TLS security implementation since last year. As a result, the previously obtained very reliable fingerprint is not valid anymore (for detailed comparisons, please refer to Figures 7 and 12). Moreover, while the fingerprints built upon the most recent datasets, namely `Campus3`, `Campus4` are consistent with one another, the cross-validation results are poor, because the resulting fingerprints correspond to typical patterns widely observed in the Internet. To conclude, application fingerprints may evolve over time and need a periodical or even constant update.

## V. Discussion

Below, we discuss the reasons for which our method results in a precise discrimination of applications.

**Incorrect/diverse implementation practices:** First, we have noticed that many protocol implementations do not follow the RFC specifications and behave slightly different from common SSL/TLS stacks. For example, in `Campus1` and `Campus2` datasets, we have observed that PayPal does not support extensions in the TLS protocol version 1.0 necessary for extensibility and security [17], [18]: the implementation simply rejects the `Client Hello` messages that contain extensions. This is the reason why, contrary to other applications, 7.1% of all PayPal sessions start (and terminate) with the `Alert` messages. A recent RFC draft discusses other incorrect practices of HTTP implementations over TLS [19]. A counterexample is a recent verified reference implementation of TLS 1.2 that supports all protocol functions, as prescribed in the RFCs [20]. Moreover, an increasing number of protocol extensions are available to expand the functionality of the TLS protocol (e.g. Heartbeat extension), which results in highly differentiated implementation practices.

**Misuse of the SSL/TLS protocol:** SSL/TLS tunneling is increasingly used as a tool for bypassing restrictions set by network configuration and security checks instead of using SSL/TSL for enforcing security. For example, since Skype uses its own security and a real-time communication protocol, SSL/TLS tunneling is only adopted to bypass network-layer firewalls. As a result, the SSL/TLS stack fingerprint is reduced to a few transitions, which significantly differs from other models.

**Server configuration:** Some SSL/TLS protocol messages are defined as optional or context-dependent. For example, in the two first datasets of our study, we have not observed any `Server Key Exchange` message in PayPal and Twitter sessions, whereas in the case of Dropbox, it always follows the `Certificate` message. The behavior depends on key exchange methods. For some of them (DHE_DSS, DHE_RSA, DH_anon), the server sends `Server Key Exchange`, because the `Certificate` message does not contain enough data to allow the client to exchange the premaster secret. However, for some other methods (RSA, DH_DSS, DH_RSA) the `Server Key Exchange` message is illegal [10].

**Application nature:** While some SSL/TLS server communication parameters can be configured and possibly changed over time, others reflect the nature of the application and depend on the service implementation. It is noteworthy that they may reflect some flow features like the session duration or the content size. For example, we can observe only few session transitions in case of Twitter, which enables its users to send short text-based messages of up to 140 characters, whereas in case of the proprietary Gadu-Gadu protocol, we may observe more than one hundred session transitions that reflect its instant messaging character (i.e., high interaction between users). Moreover, the Gadu-Gadu SSL/TLS messages are relatively short, so individual session states are composed of multiple `Application Data` messages. Contrary to Gadu-Gadu, the Skype SSL/TLS protocol messages are long and need to be divided in multiple TCP segments before sending across the network—this is another feature that cannot be easily evaded or changed with time due to the application nature.

## VI. Related Work

As new Internet applications started to use obfuscation methods (port masquerading, tunneling, packet encryption), traditional classification methods based on simple pattern matching are no more reliable. In our work, however, we demonstrate that it is possible to effectively model application flows by inspecting application layer protocols. Risso *et al.* introduced a taxonomy of payload-based classification methods [9] and argued that they are mainly based on pattern verification. We believe that a key challenge in encrypted traffic classification is to replace traditional pattern verification with more sophisticated statistical fingerprinting.

Some authors focus on classification of encrypted traffic [4], [21], [13], [22]. Bernaille and Teixeira proposed a method based on the size of the first few packets of an encrypted connection, which enables an early application recognition with the accuracy of more that 85% [4]. A more recent hybrid method tries to identify SSL/TLS encrypted application layer protocols with a combination of a signature-based and a

flow-based statistical analysis scheme [21]. Both methods are related to our proposal, however their objectives are limited either to the SSL/TLS application recognition or to classification of encrypted application layer protocols. In this work, we focus on an in-depth analysis of the SSL/TLS protocol message sequences to characterize and classify application flows. In our previous work, we considered the problem of detecting Skype traffic and classifying its service flows. We proposed a classification method for Skype traffic tunneled over TLS in addition to proprietary encryption. The method is based on the Statistical Protocol IDentification (SPID) that analyzes the distributions of flow and application layer data [13].

Bissias *et al.* presented a traffic analysis attack against encrypted HTTP streams to identify the source of the traffic by analyzing distributions of packet sizes and inter-arrival times of web requests from interesting sites [23]. Even if their work differs from our paper in terms of objectives and methodology, the conclusions remain the same: encrypting traffic does not prevent from performing some types of traffic analysis.

Lee et al. and Levillain et al. evaluated the practices of SSL/TLS servers by investigating server replies [24], [18]. They studied the details of the encryption parameters, e.g. cipher suites, key sizes, and protocol features such as supported versions and their extensions. Our work is a further step in this direction.

## VII. Conclusions

In this paper, we have proposed stochastic fingerprints for application traffic flows conveyed in SSL/TLS sessions. The fingerprints are based on first-order homogeneous Markov chains for which we identify the parameters from observed training application traces. As the fingerprint parameters of chosen applications differ considerably, the method results in a very good accuracy of application discrimination and provides a possibility of detecting abnormal SSL/TLS sessions. We have also shown that application fingerprints need to be updated periodically, because they change over time.

Our analysis of the results reveals that obtaining application discrimination mainly comes from incorrect and diverse implementation practices, the misuse of the SSL/TLS protocol, various server configurations, and the application nature. Finally, even if we are able to identify some very reliable statistical fingerprints for selected applications, it is also possible to evade the classification by avoiding implementation mistakes and building the secure layer on limited, but widely-used set of SSL/TLS states.

In the future work, we plan to investigate further the proposed method on a wider range of Internet applications and cross-validate on other heterogeneous datasets gathered in various subnetworks. We also aim at analyzing the SSL/TLS stack to verify its consistency with protocol recommendations and best security practices. Finally, we plan to apply the approach to reveal intrusions that exploit the SSL/TLS protocol by establishing suspicious, unlikely sessions.

## Acknowledgments

We would like to thank DIMACS and CCICADA for support, and Nina Fefferman for useful comments on the draft. This work was partially supported by the European Commission FP7 project INDECT under contract 218086.

## References

[1] "Internet Assigned Numbers Authority (IANA)," http://www.iana.org/assignments/port-numbers.

[2] A. W. Moore and K. Papagiannaki, "Toward the Accurate Identification of Network Applications," *Proc. of the PAM Conference*, pp. 41–54, 2005.

[3] S. Sen, O. Spatscheck, and D. Wang, "Accurate, Scalable In-network Identification of P2P Traffic Using Application Signatures," in *Proc. of the WWW Conference*, 2004, pp. 512 – 521.

[4] L. Bernaille and R. Teixeira, "Early Recognition of Encrypted Applications," *Proc. of the PAM Conference*, vol. 4427, pp. 165–175, 2007.

[5] A. W. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques," *Proc. of ACM SIGMETRICS*, pp. 50–60, 2005.

[6] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, "Network Monitoring Using Traffic Dispersion Graphs (TDGs)," in *Proc. of ACM IMC*, 2007, pp. 315–320.

[7] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," *Proc. of ACM SIGCOMM*, pp. 229–240, 2005.

[8] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices," in *Proc. of ACM CoNEXT*, 2008, pp. 1–12.

[9] F. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus, "Lightweight, Payload-Based Traffic Classification: An Experimental Evaluation," *Proc. of IEEE ICC*, pp. 5869–5875, 2008.

[10] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol, Version 1.2," RFC 5246 (Proposed Standard), August 2008.

[11] A. Freier, P. Karlton, and P. Kocher, " The Secure Sockets Layer (SSL) Protocol Version 3.0," RFC 6101 (Historic), August 2011.

[12] I. L. MacDonald and W. Zucchini, *Hidden Markov and Other Models for Discrete-Valued Time Series*. Chapman & Hall, 1997.

[13] M. Korczyński and A. Duda, "Classifying Service Flows in the Encrypted Skype Traffic," *Proc. of IEEE ICC*, pp. 1–5, 2012.

[14] R. Seggelmann, M. Tuexen, and M. Williams, " Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension," RFC 6520 (Proposed Standard), February 2012.

[15] J. Aldrich, "R.A. Fisher and the Making of Maximum Likelihood 1912-1922," *Statistical Science*, vol. 12, no. 3, pp. 162–176, August 1997.

[16] I. Drago, M. Mellia, M. Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside Dropbox: Understanding Personal Cloud Storage Services," in *Proc. of ACM IMC*, 2012, pp. 481–494.

[17] S. Blake-Wilson, M. Nystrom, D. Hopwood, and J. Mikkelsen, " Transport Layer Security (TLS) Extensions," RFC 4366 (Proposed Standard), April 2006.

[18] O. Levillain, A. Ébalard, B. Morin, and H. Debar, "One Year of SSL Internet Measurement," in *Proc. of ACM ACSAC*, 2012, pp. 11–20.

[19] A. Langley, "Unfortunate Current Practices for HTTP over TLS," Internet Draft, January 2011.

[20] K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, and P. Strub, "Implementing TLS with Verified Cryptographic Security," in *Proc. of the IEEE Symposium on Security & Privacy*, 2013, pp. 445–459.

[21] G.-L. Sun, Y. Xue, Y. Dong, D. Wang, and C. Li, "An Novel Hybrid Method for Effectively Classifying Encrypted Traffic," *Proc. of IEEE GLOBECOM*, pp. 1–5, 2010.

[22] R. Alshammari and A. Zincir-Heywood, "Machine Learning Based Encrypted Traffic Classification: Identifying SSH and Skype," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–8.

[23] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy Vulnerabilities in Encrypted HTTP Streams," in *Proc. of the 5th Int. Conference on Privacy Enhancing Technologies*, 2006, pp. 1–11.

[24] H. K. Lee, T. Malkin, and E. Nahum, "Cryptographic Strength of SSL/TLS Servers: Current and Recent Practices," in *Proc. of ACM IMC*, 2007, pp. 83–92.