

# “No NAT’d User left Behind”: Fingerprinting Users behind NAT from NetFlow Records alone

Nino Vincenzo Verde\*, Giuseppe Ateniese\*, Emanuele Gabrielli\*, Luigi Vincenzo Mancini\*, Angelo Spognardi†

\*Sapienza - Università di Roma, Dipartimento di Informatica, Rome, Italy

{verde, atenesi, gabrielli, mancini}@di.uniroma1.it

†IIT-CNR, Pisa, Italy a.spognardi@iit.cnr.it

**Abstract**—It is generally recognized that the network traffic generated by an individual acts as his biometric signature. Several tools exploit this fact to fingerprint and monitor users. Often, though, these tools access the entire traffic, including IP addresses and payloads. In general, this is not feasible on the grounds that both performance and privacy would be negatively affected. In reality, most ISPs convert user traffic into NetFlow records for a concise representation that does not include the payload. More importantly, a single IP address belonging to a large and distributed network is usually masked using Network Address Translation techniques, thus a few IP addresses may be associated to thousands of individuals (NAT’d IPs). We devised a new fingerprinting framework that overcomes these hurdles. Our system is able to analyze a huge amount of network traffic represented as NetFlows, with the intent to track people. It does so by accurately inferring when users are connected to the network and which IP addresses they are using, even though thousands of users are hidden behind NAT. Our prototype implementation was deployed and tested within an existing large metropolitan WiFi network serving about 200,000 users, with an average load of more than 1,000 users simultaneously connected behind 2 NAT’d IP addresses only. Our solution turned out to be very effective, with an accuracy greater than 90%. We also devised new tools and refined existing ones that may be applied to other contexts related to NetFlow analysis.

## I. INTRODUCTION

Tracking down individuals on the Internet is fairly simple nowadays. Governments collect a huge amount of Internet traffic to carry out mass surveillance programs and monitor terrorist organizations or suspects. Analysis and monitoring tools exist that can detect and locate anyone by simply looking at Internet traffic logs. Indeed, network traffic generated by a single user contains certain patterns that make it unique and, thus, discernible. Much work has been done in this area of research and it is now well established that network traffic acts as a biometric signature, or fingerprint, of the user that generated it. The way it works is that a classifier, based on machine learning, is first trained on the traffic generated by a certain individual to extract and learn distinctive characteristics from it, and then used to trace the same individual from his traffic produced afterwards, while surfing the Internet.

But, how effective are these classifiers? How realistic is the environment in which they operate?

Classifiers are indeed very effective, but it is often assumed that they are given as input the entire traffic, including payloads, headers, and other timing information. The reality is quite different, however. For performance reasons, the attacker cannot analyze or store the entire traffic, that can reach a

throughput higher than 2,000 Gbit/s in the case of large Internet Exchange points. However, ISPs often convert network traffic into NetFlow records for a more concise representation. These records are used to collect IP traffic statistics for data analysis and contain very little information (no payload, for example). Not all is lost, however, since it may not be difficult for an attacker to devise improved classifiers that can pinpoint individuals by just looking at NetFlow records. Even in this case though, it seems we must at least assume that users are identified by unique IP addresses. But again reality is harsher, and often users are hidden behind NAT and are all seen outside their network as a single entity, with only one IP address.

In this paper we show that, quite surprisingly, by mining solely NetFlow data belonging to an Internet Service Provider, or that of an Internet Exchange Point, an attacker is able to track users, and accurately estimate when they are connected to the network and which IP address they are using. The approach that we propose works also if the target user is hidden behind NAT. This privacy attack corroborates that massive spying activities, such as the ones performed by intelligence agencies ([28],[22]) are not only possible, but also require very limited computational effort. The fact that it is possible to trace users hidden behind a NAT by just mining NetFlow data is far from being obvious. Indeed, standard classification techniques are ineffective when naively applied in this context. Existing solutions that may work for NAT’d addresses are not designed to work with NetFlow records alone. We devised a new fingerprinting framework that overcomes these hurdles. Our system was tested by targeting users connected through their smartphone to an actually-deployed WiFi network, covering a metropolitan area and its surrounding county, that spans a region of nearly 15,000  $km^2$  and that employs more than 1,300 access points. Such a huge network serves routinely about 200,000 total users, with an average load of more than 1,000 users simultaneously connected behind 2 nat’d IP addresses only. Despite the “noise” of so many users inside NetFlow logs, targets were still fingerprinted with a very high accuracy. In all the analyzed cases, we achieved both *precision* and *recall* greater than 90% (more on this later). Even though we experimented with individuals using smartphones, our approach may be generalized and applied to fingerprint users carrying a variety of other devices, such as laptops or tablets.

**Application scenarios.** Some examples of applications of our solutions include:

1. *After-the-fact forensic analysis.* Since ISPs routinely collect NetFlow records, they might provide assistance to law enforce-

ment agencies and track down criminals after the fact or learn about their habits at different points in time.

2. *Covert intelligence operations.* Officials may be mandated to trace certain individuals on national security grounds. While ISPs readily collaborate with their own Governments, they may refuse to provide relevant information to other entities outside the borders. Sometimes even asking for an authorization to access data may not be an option. Our framework can easily be used to monitor the traffic on out-of-bounds routers and fingerprint individuals roaming within the domains of targeted ISPs.

If abused, our solutions can also be used to violate the privacy of citizens. Therefore, in this respect, our work should be interpreted as a warning sign of what certain organizations might be able to undertake.

**Contributions.** Our main contribution is to provide the first solution for fingerprinting individuals hidden behind a NAT router when only NetFlow records are available. We believe our solutions are significant since we deployed them within an existing metropolitan WiFi network with thousands of (real) users. Along the way, we also devised novel techniques and classification strategies that required appreciable efforts and which we believe are of independent interest (at least to those working with NetFlow analysis). In particular, other relevant contributions are:

1. A novel method to encode NetFlow records into training sets suitable for several HMM classifiers run in parallel. We employed HMMs for their ability to properly capture “time” information, as in time series analysis, and to handle changes in data distribution over time.
2. The definition of a *User Detector* that operates on the outputs of multiple HMMs, and performs a time interval aggregation of the individual results. In our experiments, a simple weighted sum was adopted as aggregation function for its ability to meaningfully combine distinct HMM outputs (more on this later).
3. The design of a classification component, based mainly on Random Forest, that can automatically interpret the output of the gating component and supply the final classification.

**Organization of the paper.** Section II reports on related work. In Section III, we introduce the background and the definitions needed to introduce our framework. The different components of the framework are detailed in Section IV, while Section V reports several experimental results that show the viability of our approach. Finally, Section VI provides the conclusions.

## II. RELATED WORK

Our main claim in this paper is that there is no effective technique to fingerprint individuals hidden behind NAT when only NetFlow records are available. To justify this claim, we performed an extensive and rigorous analysis of previous proposals in the area and classified them based on (1) the intended target (whether device, host, user, application, etc.) and (2) the technique used to perform the fingerprinting/profiling. Indeed, it is important to remark that there are many proposed solutions that work for, e.g., devices or applications but cannot be used for individuals. Or, rather, they work for individuals but only when payloads are available (but fail when applied to NetFlow logs only).

TABLE I. CLASSIFICATION OF THE RELATED WORK

Target	Used Technique		
	NetFlow Analysis	Stream Statistics analysis	Payload Analysis
User and Host Profiling	[24], [23]	[37], [38], [35], [4], [18], [26]	[39], [13]
Traffic and Application Profiling	[3], [27], [31]	[40], [25], [17], [32]	[6]
Information Leaks		[21], [33], [5], [36]	
Device Profiling		[19], [12], [8]	

We report the results of our classification in Table I. We identified three main strands: works based on *Netflow analysis*, those based on *stream statistic analysis*, and those based on the *payload analysis*. With respect to the *target*, we identified four different categories: *user and host profiling*, *traffic and application profiling*, *information leaks* and *device profiling*.

a) *User and Host Profiling:* We distinguish two additional sub-categories: behavioral targeting and single user/host profiling. We anticipate, however, that behavioral targeting focuses only on identifying communication patterns that are in common to many users (or hosts), with the intent to analyze anomalies. Thus, it does not consider the same problem addressed in this paper.

Behavioral targeting refers to a range of technologies and techniques used to model the behavior of sets of users or hosts with the intent of identifying (1) common features, or (2) anomalies from the typical behavior. The main objective of this line of work is to build user profiles to offer customized services and it is thus valuable to website publishers and advertisers. As described in [39], some of these techniques analyze the interactions between users and one or more federated content provider servers. Identifying groups of Internet hosts with a similar behavior is convenient to detect security breaches, such as DDoS, worms, viruses, botnets, etc. In [37] and [38], the authors profiled the Internet backbone traffic with the intent of discovering compromised hosts. They analyzed the communication patterns of end-hosts and services via data mining and information-theoretic techniques. In the end, they showed how to identify common traffic profiles, as well as anomalous behavior patterns, that are of interest to network operators and security analysts. A similar problem was addressed in [35], where host profiles were used to detect anomalous behaviors during the Slammer worm spread. We emphasize that our target is different: we are not interested in creating a cluster of similar users but rather our focus is to improve our ability to single out users.

Melnikov et al. [24] introduced a proof-of-concept technique used to distinguish between users by analyzing their NetFlow traffic. However, details on how their technique can be employed in a real-world environment are missing and their experiments provide very limited insights. In [4], the authors show how to distinguish distinct users during their online playing activities. However, their approach is not generalizable and hence not applicable to our context. Profiling end-host systems based on their transport-layer behavior was proposed in [18]. There, the authors used graphlets to capture information flows and inter-flow dependencies. Their basic technique though cannot be used to re-identify users later on. Re-identification is explicitly considered in [13], where patterns mined from web traffic are used to link multiple sessions of the same user. Un-

fortunately, this solution does not work for users hidden behind NAT. The same applies for the work in [23], where NetFlows are used to detect behavioral changes, identify anomalies and trace the propagation of malware. In [26], Pang et al. showed that users can be tracked through implicit identifiers in 802.11 networks even when unique addresses and names are removed. However, their technique works only when the attacker is physically close to the target and can eavesdrop the wireless traffic. A completely different approach is proposed in [34], where the authors use Open Source Intelligence to improve the results of host profiling. This refinement can also be used within our framework to improve our results.

*b) Traffic and Application Profiling:* This category aims at recognizing either the type of network traffic (i.e., p2p, streaming video, VoIP, etc.), or the application that generated it (i.e., Skype, Gnutella, etc.). It is a classification technique employed by network administrators to monitor network traffic and identify different applications. As shown in Table I, several works rely on statistics extracted from the network traffic. To compute these statistics, it is sometimes sufficient to access just the header of TCP/IP packets [40], [25], [17], [32], but in general the entire packet payload is necessary [6]. These approaches do not scale for large networks where NetFlow analysis is preferred. Indeed, in [3], [27] and [31], the authors use NetFlow analysis for traffic monitoring and application classification. However, their techniques cannot be applied to fingerprint users hidden behind a NAT router.

*c) Information Leaks:* *Information leaks* category aims at analyzing **encrypted** traffic with the intent to uncover any useful information. In [21] and [33], two attacks against encrypted HTTP streams were presented. The authors identified webpages visited by victims based on a pre-built database containing webpage fingerprints. Chen et al. [5] make use of fingerprints of different webpages to infer users' browsing habits. Similar approaches have been used to identify spoken phrases within a VOIP call. In [36], for example, the authors show how the lengths of encrypted VoIP packets can be used to identify spoken phrases of a variable bit rate encoded call.

*d) Device Fingerprinting:* it aims at identifying different physical devices (such as mobile phones). Microscopic deviations in hardware components, such as the clock skews, were used in [19] to fingerprint computer devices. This technique works even when the device is behind a NAT or a firewall, and also when the device's system time is maintained via NTP or SNTP. Clearly, it won't work to distinguish two devices of the same model (i.e., with the same hardware specs). The same issue applies to the work in [12], where a technique to accurately identify the software driver used by 802.11 wireless adapters is proposed. In [8], the authors propose to identify devices by using timing analysis of probe request frames emitted from wireless client stations when scanning for access points. However, this approach cannot be used at the network level since wireless frames can only be captured within the wireless transmission range.

Researchers are also proposing several possible countermeasures to actively mitigate the information leakage, with privacy-preserving [14] and obfuscated [30] data.

### III. BACKGROUND AND DEFINITIONS

This section summarizes two main components used in this work: the NetFlow technology and our machine learning approach based on HMMs. Netflows are used as building blocks to enable the analysis of the network traffic. HMMs are adopted to model the traffic of a user and to re-identify him later on, when his traffic will be mixed with the traffic of a number of other users.

#### A. NetFlows

NetFlow is a protocol designed by Cisco to collect IP traffic information while getting rid of any IP packet payload. It makes use of compact representations of the packet exchange between two network peers. NetFlow collects traffic information but discards IP packet payloads and it is commonly used for network traffic monitoring and reporting. NetFlow v9 has evolved as a IETF standard called IPFIX, already implemented by most network equipment vendors [1].

We decided to focus on NetFlow since most routers natively support it and it is the de facto standard for a compact representation of a large amount of network traffic. Furthermore, NetFlow records do not contain packet payloads, thus the activity of collecting and analyzing them is considered legitimate and does not raise privacy concerns (unlike deep packet inspection).

A NetFlow enabled device (a *probe*) extracts from each packet a key composed of specific IP header fields. To simplify the exposition, we can think to this key as a 5-tuple containing IP source and destination addresses, source and destination ports, and the protocol used. More formally, we define a NetFlow key function that takes as input an IP packet and outputs a 5-tuple of attributes.

*Definition 1:* A **NetFlow key function**  $f_{key}$  is defined as  $f_{key} : \mathcal{I} \rightarrow \mathcal{K}$ , where the set  $\mathcal{I}$  denotes the set of possible IP packets, and the set  $\mathcal{K}$  is the set of 5-tuples of the form:  $(IP_{src}, port_{src}, IP_{dst}, port_{dst}, protocol)$ <sup>1</sup>

A NetFlow probe applies the NetFlow key function to each single packet and dynamically builds in its cache memory a set of *NetFlow raw records*. These also contain several other attributes, among which the most relevant are: cumulative number of exchanged packets, bytes counters, flow starting and finishing timestamps, TCP flags, and Type of Service (ToS). More formally:

*Definition 2:* A **NetFlow raw record**  $nfr$  is composed of a key value  $k \in \mathcal{K}$  and a data tuple  $(packets, bytes, start\_timestamp, end\_timestamp, TCP\_flags, ToS)$ . Each element of the data tuple represents a feature of the set of IP packets  $I \in \mathcal{I}$ , with  $k = f_{key}(I)$ , exchanged within a single connection between two network peers.

NetFlow raw records, along with the output of the NetFlow key function, are subsequently sent via UDP to a NetFlow collector, for storing and analysis purposes. A new  $nfr$  is sent to the collector when the connection is closed (i.e., a packet

<sup>1</sup>NetFlow v5 includes two more fields within the key but they are not relevant to our work. These are: the type of service and the index of the ingress interface. Newer NetFlow versions have also introduced the possibility to specify custom keys and attributes.

explicitly terminates the flow via TCP FIN or RST) or the NetFlow expires. Indeed, a NetFlow can expire for three main reasons: (1) the flow has been inactive for a time period longer than the *inactive timeout*; (2) the flow has been active for a time period longer than the *active timeout*; (3) the flow cache is full and some space needs to be freed for new flows. Default values for inactive timeout and active timeout are set to 15 seconds and 30 minutes, respectively.

The NetFlow collector may store multiple records for each NetFlow key. Indeed, the router may receive a packet with the same  $f_{key}$  of an expired NetFlow raw record. In this case, a new  $nfr$  with the same key is created. We will leverage this feature to build our framework later. First, we need to define the *flow* as the composition of several related NetFlow raw records.

**Definition 3:** A **flow**  $f$  is defined as a set of NetFlow raw records  $\{nfr_1, \dots, nfr_n\}$  such that  $\forall 1 \leq i, j \leq n$ , the key associated with  $nfr_i$  is equal to the key associated with  $nfr_j$ .

We realized that the flow defined above is very convenient and effective in identifying users. For instance, a flow properly captures certain usage patterns and is oblivious to NAT routers. Indeed, two NAT'd users connecting to the same IP address and port will be assigned two distinct local ports. Therefore, two distinct flows will be generated, one per each user.

In the following, we define a *bi-directional flow* as the union of two distinct flows:

**Definition 4:** Given an IP protocol *protocol*, two pairs of IP addresses and ports  $ip_1:port_1$  and  $ip_2:port_2$ , we define a **bi-directional flow** as the union of the flows from  $ip_1:port_1$  to  $ip_2:port_2$  with the one from  $ip_2:port_2$  to  $ip_1:port_1$ .

Finally, we will consider only *ordered flows*, by applying a ordering function *sort* that rearranges a flow, sorting its  $nfr$ 's with respect to the start timestamp. More precisely, we say that an **ordered flow** is the flow obtained by applying the *sort* function to the  $nfr$ 's that compose the input flow, namely  $sort(f) = \{nfr_1, \dots, nfr_n\}$ , such that  $\forall i < j$ ,  $nfr_i.start\_timestamp \leq nfr_j.start\_timestamp$ .

Similarly, we can say that an **Ordered Bi-directional Flow (OBF)** is the bi-directional flow obtained by applying the *sort* function. OBFs are therefore sequences of NetFlow raw records that describe how the connection between two endpoints evolved over time. OBFs are essential components in our framework. When properly encoded, they are used to train HMM classifiers which are described next.

Table II reports an example of generation of OBFs. Five Netflow raw records are listed as  $nfr_1$  to  $nfr_5$ . Each one is composed of a key and a data tuple. The key contains source and destination IP addresses, ports, and protocol. The data tuple associated with the key reports the number of exchanged packets, bytes, the start and end timestamps, TCP flags and the type of service. Since  $nfr_1$  and  $nfr_3$  share the same key value, they are aggregated together in the same flow. As such, the five netflow raw records form four different flows  $f_1, f_2, f_3, f_4$ . Netflow raw records  $nfr_1, nfr_2$  and  $nfr_3$  form an Ordered Bidirectional Flow. Indeed, they are related to the same TCP connection between the IP addresses 192.168.1.10 and 173.124.18.52. Netflow raw records  $nfr_5, nfr_4$  form another Ordered Bidirectional Flow. Note that  $nfr_5$  precedes  $nfr_4$

since the start timestamp of  $nfr_5$  precedes the start timestamp of  $nfr_4$ .

## B. Hidden Markov Models

An HMM is a Finite State Machine able to model a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce a sequence of observed symbols, or vectors, as in our case [29]. We consider HMMs with continuous observation densities. In its compact form, an HMM is defined by  $\lambda = (A, B, \pi)$ :

**A:**  $N \times N$  state transition probability matrix, where  $N$  is the number of hidden states. Each matrix element  $a_{ij}$  is the probability of a transition of the hidden process from state  $i$  to  $j$ .

**B:**  $N$  observation probability densities. Each probability density represents the probability of a certain observable, when the hidden process is in state  $i$ .

**$\pi$ :**  $N$  length initial state probability vector. Each vector element  $\pi_i$  is the probability that the hidden process starts from state  $i$ .

The use of HMMs is proposed when solving one or more of the following problems related to the modeled phenomenon [29]:

**Problem 1:** Given an HMM  $\lambda$  and a sequence  $O$  of observables  $o_1, o_2, \dots, o_t$ , find the probability  $P(O|\lambda)$  that these observables are generated by the given model.

**Problem 2:** Given a model  $\lambda$  and a sequence  $O$  of observables  $o_1, o_2, \dots, o_t$ , find the sequence  $Q$  of states  $q_1, q_2, \dots, q_s$  that maximizes the probability  $P(O|\lambda)$ .

**Problem 3:** Given  $N$  the number of states, the initial state probability distribution  $\pi$ , a sequence  $O$  of observables  $o_1, o_2, \dots, o_t$ , and (if known) the corresponding sequence  $Q$  of states  $q_1, q_2, \dots, q_t$  that emitted them, find the model  $\lambda$  that maximizes  $P(O|\lambda)$ .

To solve the third problem, an iterative procedure is used, that learns the model by adjusting its parameters and optimally adapting them to the observed training data. The training process is able to create the best model for the observed phenomenon. The learning is said supervised when the model can be trained with the knowledge of both the emitting states and the observables, otherwise it is said unsupervised.

Hidden Markov Models (HMMs) are widely used in sequences analysis since there exist efficient algorithms to solve the three problems defined above [9], [15], [20]. In this paper, HMMs are employed to model and recognize user traffic. In this case, the observables are the NetFlow raw records: as described in Section III-A, each  $nfr$  is represented by a data tuple of  $t$  values, corresponding to its  $t$  attributes (number of packets, start timestamps, etc.). In particular, we consider that the observables are  $t$  dimensional vectors distributed according to  $N$  multivariate Gaussian distributions, one for each state: we will adopt one  $N \times t$  matrix, containing the means, and  $N$  covariance matrices, to define the  $t$ -dimensional multivariate Gaussian distributions. In other words, for each state of the model, we have  $t$  Gaussian densities, one for each of the  $t$  vector elements and, to represents such densities, we have to specify their means and their covariances.

TABLE II. ORDERED BIDIRECTIONAL FLOWS: A SIMPLE EXAMPLE

Object	Name	Content
Netflow Raw Records	$nfr_1$	key: (192.168.1.10, 5430, 173.124.18.52, 80, TCP), data tuple: (145, 1815, 1375690161, 1375699541, SIN+ACK, 0)
	$nfr_2$	key: (173.124.18.52, 80, 192.168.1.10, 5430, TCP), data tuple: (5, 421, 1375690290, 1375699650, SIN+ACK, 0)
	$nfr_3$	key: (192.168.1.10, 5430, 173.124.18.52, 80, TCP), data tuple: (12, 1815, 1375690690, 1375699703, SIN+ACK+RST, 0)
	$nfr_4$	key: (192.168.1.10, 2345, 64.12.121.12, 443, UDP), data tuple: (1, 196, 1375690600, 1375699705, SIN+ACK, 0)
	$nfr_5$	key: (64.12.121.12, 443, 192.168.1.10, 2345, UDP), data tuple: (1, 12, 1375690590, 1375699596, FIN, 0)
Flows	$f_1$	$\{nfr_1, nfr_3\}$
	$f_2$	$\{nfr_2\}$
	$f_3$	$\{nfr_4\}$
	$f_4$	$\{nfr_5\}$
OBFs	$OBF_1$	$(nfr_1, nfr_2, nfr_3)$
	$OBF_2$	$(nfr_5, nfr_4)$

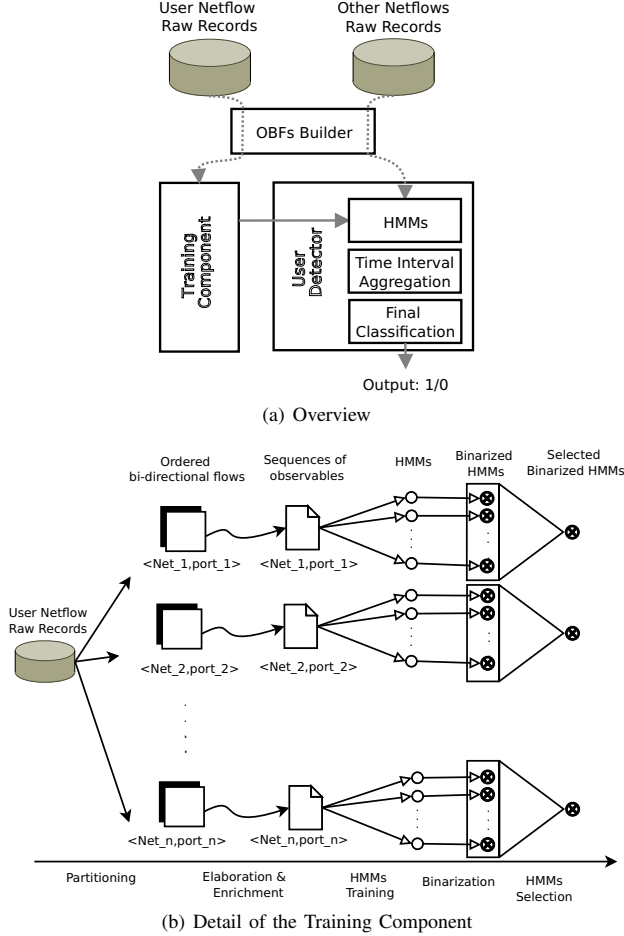


Fig. 1. Our Fingerprinting Framework.

To realize our framework, we will make use of HMMs to learn the network traffic profile of a target user. The training phase will be carried out by solving an instance of the Problem 3 above, via an unsupervised approach. Then, in the classification phase (when users are subsequently recognized), trained HMMs will essentially solve instances of the Problem 1 above.

#### IV. THE FINGERPRINTING FRAMEWORK

Our proposed fingerprinting framework has two main components: the Training Component and the User Detector. The

first component operates as follows: (1) It takes as input NetFlow raw records of the target user, (2) trains a set of HMMs to recognize its OBFs, and (3) selects the HMMs that achieve the best performance. The User Detector operates as follows: (1) It uses the selected HMMs to classify unknown traffic, (2) aggregates the results into a new dataset that describes time intervals, and (3) applies a final classification to the aggregated dataset. At the end of this process, the user detector will determine whether, during a time interval, the network traffic contains anything from the target user. More details on these two components are provided next.

##### A. Training Component

This component is tasked with creating a set of HMMs, collectively able to recognize the traffic of the target user. We adopted an approach to exploit multiple learners, called mixture of experts (ME) [41]. Unlike typical ensemble methods (where individual learners are trained for the same problem), a mixture of experts works in a divide-and-conquer strategy, where a complex task is broken up into several simpler and smaller subtasks on which individual learners (the *experts*) are trained.

In our case, we use a natural task repartition since the experts are several specialized HMMs, each dedicated to recognize user traffic towards a single network service. Figure 1(b) graphically describes the process of realizing trained HMMs in our framework. The entire process is divided into five sub-phases: Partitioning, Elaboration and Enrichment, HMMs Training, Binarization, HMMs Selection.

*e) Partitioning phase:* We start with a collection of NetFlow raw records  $nfrs$  that are generated by user  $U$ . In the partitioning phase, these  $nfrs$  are divided and organized in subsets. Each subset is related to a service or to a set of services accessed by  $U$ . As in [26], we assume that the IP addresses contacted by the user  $U$ , along with the corresponding ports, are *implicit identifiers* of the accessed services. However, we must take into account that IP addresses may vary at each service request. For example, it is very likely to contact two distinct IP addresses when accessing *www.youtube.com* twice even in a short time interval. To overcome this problem, we use the *whois* lookup protocol [7] to map each IP address to a netrange that identifies the IP block of addresses that the service provider controls. In particular, we select the smallest IP address range returned by querying five Regional Internet Registries, namely *arin*, *ripe*, *apnic*, *iana* and *lacnic*. Combining the retrieved netrange with the contacted port, we obtained a pair  $\langle netrange, port \rangle$ , that constitutes the key

used to partition the collection of the NetFlow raw records of the target user. The last step is the organization of the  $\langle \text{netrange}, \text{port} \rangle$  subsets in OBF sets. This task is carried out by the *OBFs builder* that combines and sorts the *nfr*'s with the same  $\langle \text{netrange}, \text{port} \rangle$ . This results in several sets of OBFs organized by  $\langle \text{netrange}, \text{port} \rangle$ . Each element of a set represents a sample of the connections between the target user  $U$  and the related service. It will become an observation sequence given as input to an HMM.

*f) Elaboration and Enrichment phase:* In this phase, the OBF elements of the above partitioning are transformed into the observables that will ultimately be used by the HMMs. In particular, the features of each single *nfr* of a OBF become the elements of a feature vector. Therefore, each single OBF becomes a sequence of feature vector sequences, namely the observation sequences of the HMMs. The feature vectors are composed of a combination of the following OBF features:

**Gap:** counts the milliseconds elapsed between a *nfr* and the previous one in the OBF. It is set to 0 for the first *nfr*.

**Packets:** is the number of packets reported by a *nfr*.

**Bytes:** is the number of bytes reported by a *nfr*.

**Direction:** is the direction of a *nfr*. This value is equal to 1 if the related connection was outgoing (originated from the target user), 0 if it is ingoing (originated from the other end-point).

We considered such simple features because they convey the main characteristics of the exchanged traffic. Indeed, studies like [10] have proven that by using only coarse features such as total upstream and downstream bandwidth, it is possible to infer private and sensitive information from the encrypted network traffic.

*g) HMMs Training:* The sequences of feature vectors are used to train multiple HMMs in parallel. For each subset of OBFs, we use 80% of observation vectors to train the HMMs, while we save the remaining 20% for subsequent phases. Several HMMs are trained by varying the number of states and the subset of features of the observation vectors. In particular, we used two, three and four states with 14 different combinations of features, namely:  $\{\text{Pkts}\}$ ,  $\{\text{Bytes}\}$ ,  $\{\text{Gap}\}$ ,  $\{\text{Direction}\}$ ,  $\{\text{Pkts}, \text{Bytes}\}$ ,  $\{\text{Pkts}, \text{Gap}\}$ ,  $\{\text{Gap}, \text{Bytes}\}$ ,  $\{\text{Direction}, \text{Bytes}\}$ ,  $\{\text{Direction}, \text{Pkts}\}$ ,  $\{\text{Direction}, \text{Gap}\}$ ,  $\{\text{Gap}, \text{Bytes}, \text{Pkts}\}$ ,  $\{\text{Direction}, \text{Bytes}, \text{Pkts}\}$ ,  $\{\text{Direction}, \text{Gap}, \text{Pkts}\}$ ,  $\{\text{Direction}, \text{Bytes}, \text{Gap}\}$ . These 14 combinations are all the possible subsets of cardinality at most 3, that can be achieved starting from the set that contains the 4 features:  $\{\text{Pkts}\}$ ,  $\{\text{Bytes}\}$ ,  $\{\text{Gap}\}$ ,  $\{\text{Direction}\}$ . We set the initial parameters of each HMM via the K-Means algorithm [16]. The outcome of this phase consists of 42 uniquely trained HMMs for each OBF subset created in the previous phases.

*h) Binarization phase:* Each trained HMM is able to evaluate a sequence of feature vectors and to give the probability that such a new observation was obtained capturing the traffic of the target user with a given service. This is the behavior of a probabilistic classifier. The next step is to set a probability threshold to achieve a binary classifier that recognizes only two classes: if the observed sequence has a probability lower than  $t$ , it will be classified as 0, otherwise it will be classified as 1. The threshold  $t$  is chosen for each HMM by testing the 20% of observation vectors set aside during

the training phase, mixed with other observation vectors not belonging to the target user  $U$ . In particular, the threshold is set as the value that maximizes the accuracy in terms of balanced F-measure  $F_1$ , namely

$$F_1 = \frac{2PR}{P + R}$$

where  $P$  is the *precision* and  $R$  is the *recall*: precision is the ratio between the positively-and-correctly classified samples and the positively classified samples, whereas the recall is the ratio between the positively-and-correctly classified samples and all the positive samples considered in the test [2].

*i) HMM Selection:* During this last phase, the HMM with the best accuracy is selected. Namely, the HMM that maximizes the F-measure is chosen among all 42 possible HMMs available. The corresponding accuracy (in terms of F-measure) will be subsequently used to assign a weight to the HMM, during the user detection process.

## B. User Detector

The goal of the Training Component (described previously) is to release a series of trained HMMs, each of them specialized in recognizing the traffic of the target user related to a unique network service. The User Detector employs these HMMs to analyze some collected traffic and to determine whether there is any information of the target user. It aggregates the results by time intervals, and applies a final classification to the aggregated dataset.

*j) HMM Classification:* The User Detector component starts with a collection of NetFlow raw records. The OBFs Builder is used to combine the *nfr*s in OBFs. Given a new OBFs, the corresponding HMM is selected, the required features are extracted, and the result of the classification is stored for further computations. If there is no specialized HMM for a new OBF, then it can be discarded as soon as it arrives. This is because there is no way, in this case, to determine whether it was generated by  $U$ . Classification results are stored to be later retrieved and aggregated with the ones from the same time interval. In particular, for each single OBF, we store the predicted class (i.e., 1 or 0) and the index of the HMM used for the classification.

*k) Time Interval Aggregation:* Once all the OBFs of the time interval have been classified, we use an aggregation function to summarize the results. The aggregation will generate a concise record whose length depends on the number of HMMs trained for the target user  $U$ . The record will represent the weighted number of OBFs that each HMM has attributed to the user  $U$  during the time interval. Therefore, a record will be of the form:

$$\langle \text{weight}(\text{HMM}_1) \times \text{OBF}_1(U); \dots; \text{weight}(\text{HMM}_n) \times \text{OBF}_n(U) \rangle$$

where  $\text{weight}(\text{HMM}_i)$  is the weight assigned to the  $i^{\text{th}}$  HMM (that is its accuracy),  $\text{OBF}_i(U)$  is the number of OBFs recognized by  $\text{HMM}_i$  as belonging to the user  $U$  during that interval, and  $n$  is the number of trained HMMs.

l) *Final Classification*: The User Detector has to finally associate a binary label to the record generated after the Time Interval Aggregation: 1 indicates that the user was connected to the network during that time period, 0 otherwise. A naive solution would be to sum all the values composing a single record, and fix a threshold to convert the sum into a binary result. However, better results can be attained by using standard classification algorithms applied on records from several time intervals. In the experiments section, we will compare the performance of several classification algorithms, such as Support Vector Machine, Random Forest, JRip, Multilayer Perceptron and Naive Bayes. In the end, this approach turns out to be very effective, providing a very high *precision* and *recall*.

## V. EXPERIMENTS AND DISCUSSION

In this section we briefly describe the implementation of the proposed fingerprinting framework and how the experiment environments were set up. Then, we discuss and report on the results we attained.

### A. Framework implementation

To prove the effectiveness of our proposal, we realized a working prototype of our fingerprinting framework, as designed in Figure 1(a). The core system has been coded within a Java environment. In particular, both the training component and the user detector component have been implemented in Java, and a Java/R Interface called JRI<sup>2</sup> has been used to run R from the Java application. Basically, HMMs have been implemented in R, with the support of the package *mhsmm*<sup>3</sup>. This package provides parameter estimation and prediction for HMMs for data with multiple observation sequences, and supports Multivariate Gaussian distributions. The Final Classification phase has been realized by using the Weka library, that provides a large suite of machine learning algorithms. In the following, the experiment environments and the attained results are discussed in details.

### B. Experiment environments

The typical hardware configuration to keep track of NetFlow generated by user traffic is composed of one or more routers and switches with NetFlow capabilities (the *probes*) and a NetFlow collector. During its normal duty, each NetFlow-enabled device (router or switch) creates and elaborates NetFlow records for every single packet. Then, it transmits batches of collected records to the collector that stores them for further analysis. We set up two different environments where we collected NetFlow traces from real users, without sampling, to realize both a small scale and a large scale experiment.

For the small scale experiment, we configured a NetFlow-enabled router and a single wireless access point to create a WiFi network within our premises. The router was the default gateway for the network and was configured to send NetFlow data to a local collector. We profiled a total of 26 different users accessing the Internet with their mobile devices during one month of monitoring, collecting 4.8 GB of traffic and more than 500,000 NetFlow raw records.

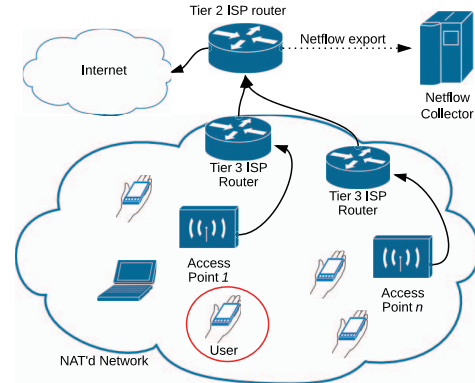


Fig. 2. Large Scale Experiment Network

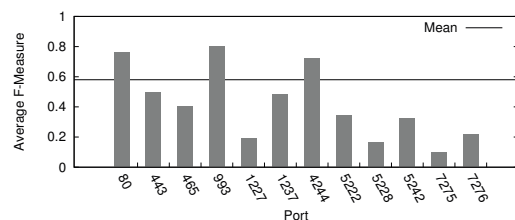


Fig. 3. Ordered Bi-directional Flows classification. Average accuracy by port number

For the large scale experiment, we had access to the NetFlow traffic of a metropolitan network that provides public WiFi connectivity to a large region with 200,000 registered users, covering 144 towns in an area spanning nearly 15,000  $km^2$ , and including about 1,300 wireless access points (Figure 2 outlines the network configuration). A NetFlow probe was placed inside the ISP that provides connectivity services for the public WiFi network. The public WiFi network is NAT'd and uses only 2 public IP addresses with an average of 250 GB of daily traffic, thus producing nearly 20 million NetFlow raw records per day.

### C. Small Scale Experiment

We monitored the Internet connection of 26 different users for one month, collecting their traffic when they were connected to our access point. The first week's NetFlow data was used for the Training Component, the rest to train and test the entire framework through cross-fold validation. During the training week, the smartphones of the 26 target users contacted a total of 74 different services (composed of *netrange*, port pairs). Figure 3 reports the average accuracy, in terms of F-measure, of the HMMs trained to recognize the traffic toward these services aggregated by port. It can be seen that HMMs assigned to recognize the traffic toward port 80 have an average accuracy of 0.76, while those assigned to recognize the traffic toward port 443 (which is encrypted) have an accuracy of 0.49. It may seem that the encryption reduces the accuracy. However, this is not completely accurate. Indeed, the features that we use to train our classifiers with (such as the gap or the packets transmitted) are not significantly influenced by the encryption. This is also confirmed by the results achieved analyzing the encrypted traffic flowing in port 993 (IMAP over TLS/SSL).

<sup>2</sup><http://www.rforge.net/JRI/>

<sup>3</sup><http://cran.r-project.org/web/packages/mhsmm/index.html>



TABLE III. SMALL SCALE EXPERIMENT—TIME INTERVAL CLASSIFICATION. COMPARISON OF FIVE DIFFERENT CLASSIFICATION ALGORITHMS USED IN THE FINAL CLASSIFICATION PHASE. AVERAGE VALUES ARE REPORTED.

Algorithm	TPR	FPR	Prec	Recall	F-measure
Rand. Forest	0.95	0.07	0.95	0.93	0.94
Naive Bayes	0.55	0.08	0.55	0.87	0.67
MLP	0.86	0.14	0.86	0.88	0.87
SVM	0.69	0.09	0.69	0.88	0.76
JRip	0.94	0.12	0.94	0.89	0.91

Indeed, the HMMs on this port have accuracy equal to 0.80 on average. On the other hand, port 7275 has very low accuracy. This port is related to the Open Mobile Appliance User Plane Location protocol that is used by mobile devices to receive GPS info quickly. Thus, the traffic to this port is somehow automatic and this explains the low accuracy we measured. In general, certain services are well suited for identifying users while others are more “impersonal”. This is confirmed also by Figure 4. It details the accuracy of all HMMs employed by the User Detector to recognize the traffic of a specific user (we selected the worst performing). Notice that services hosted by Google, and accessed through ports different than 80, have a fairly low accuracy. On the other hand, the Amazon Elastic Compute Cloud (Amazon EC2), a service used by many application developers, reaches very high accuracy.

Services that have a higher accuracy are the ones that should be used to fingerprint individuals. This is why, in the time interval aggregation, the accuracy is represented by a weight. Note also that several services are very popular while others are used by very few users (who are then easier to profile). For instance, the service DNSINC-3 in the figure is related to an Android application called *DynDNS client* from Dynamic Network Services, Inc., and that application amounts to less than 50,000 downloads worldwide.

Other than the accuracy of the OBFs classification, we must also measure the overall performance of the framework to determine when the target user is connected. For this, we tested several classification algorithms for the final classification phase. Table III sums up the results that we achieved by using five different algorithms: Random Forest, Naive Bayes, Multilayer Perceptron (MLP), Support Vector Machine (SVM) and JRip (we used the Weka implementation of these algorithms). Random Forest behaved better than the rest in all the evaluation metrics that we considered. It reaches 95% of true positive rate, and only 7% of false positive rate. Precision, Recall and F-measure (i.e., the harmonic mean of precision and recall) are equal to 0.95, 0.93, 0.94, respectively.

The area under the ROC (Receiver Operating Characteristic) is a convenient way of comparing classifiers. A random classifier has an area of 0.5, the ideal one has an area of 1. Under this metric, we confirmed that Random Forest performs better than other algorithms within our framework (SVM was the worst) [11]. Indeed, Table IV reports the average ROC Area related to 6 different users selected among the 24 that we profiled (individual values are also reported): for all users, Random Forest reaches a ROC area close to 1 (from 0.95 to 0.97) and it has the lowest variance as well. The latter reveals that Random Forest is appropriate for classifications involving distinct users, with only a small variation over the

final performance.

#### D. Large Scale Experiment

For the large scale experiment, we used the large WiFi network described in Section V-B. Consider a scenario in which an intelligence agency intends to monitor and trace a group of, say, five suspects. The agency, with no wiretap warrant or direct access to the large WiFi network, can only export all the NetFlow data produced by the main ISP router providing connectivity to the large WiFi network (see Figure 2). For the sake of the experiment, we enrolled five volunteers that used their own mobile phone, without installing new applications or changing their usual behavior. Two of them had a pristine phone with no third-party applications (a worst-case scenario for our profiling framework), but with certain services correctly configured, such as (1) Gmail, (2) Twitter, (3) Facebook, (4) Skype, in addition to (5) the backup of the phone camera (via Dropbox). All five volunteers were previously profiled for a period of 8 hours by inducing them to connect to an access point directly controlled by us (acting as the intelligence agency). The average training traffic collected per user was of 12.64 MBytes.

During the test phase, an average of around 1,100 users were connected to the WiFi network simultaneously, and all of them were NAT'd behind only two IP addresses. During the experiment, we attempted to detect the presence of each one of the suspects by mining the NetFlow data. The suspects used the WiFi network only during specific hours, from 10AM to 10PM. Approximately 100 million NetFlow raw records were analyzed during the test phase that lasted 24 hours. This roughly corresponds to 96 GB of traffic, with an average of 9 Mb per second (Mbs). About 700,000 unique netrangles were contacted by the 1,100 users. Only 0.02% of these netrangles were contacted during the training phase by the suspects. We had no prior knowledge about any user, but our 5 volunteers.

Table V shows the results achieved. Note that we report only on the results achieved with the Random Forest classifier since it is the best performer in the final classification step. In all cases, the true positive rate is higher than 0.9, while the false positive rate is lower than 0.08. Furthermore, *precision* and *recall* are higher than 0.9 in all cases. *Suspect 1* was successfully detected, while *Suspect 2* and *Suspect 4* produced one false positive and one false negative only.

*Suspect 3* and *Suspect 5* had pristine phones, using only applications installed by the phone manufacturer, as mentioned above. These two users were misclassified for the first two hours but then recognized successfully thanks to their distinct application usage patterns, i.e., users started using the same services but in increasingly distinct ways over time. However, considering that the training period of the large- and the small-scale experiments are noticeably different (8 hours and one week, respectively), another interesting aspect we are planning to investigate in the future is to determine to what extent the training time affects the detection accuracy.

## VI. CONCLUDING REMARKS

We showed that it is possible to fingerprint NAT'd individuals when only NetFlow records are available. This is a very realistic scenario since most networks are NAT'd and





- [7] L. Daigle. RFC 3912, WHOIS Protocol Specification, Internet Engineering Task Force. Technical report, 2004.
- [8] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee. Identifying unique devices through wireless fingerprinting. In *Proceedings of the first ACM conference on Wireless network security - WiSec '08*, page 46, New York, New York, USA, 2008. ACM Press.
- [9] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge Univ. 1999.
- [10] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 332–346, Washington, DC, USA, 2012. IEEE Computer Society.
- [11] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories, 2004.
- [12] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. *Proceedings of the 15th conference on USENIX Security Symposium*, 15:167–178, 2006.
- [13] D. Herrmann, C. Gerber, C. Banse, and H. Federrath. Analyzing Characteristic Host Access Patterns for Re-Identification of Web User Sessions. In *Proceedings of the 15th Nordic conference on Information Security Technology for Applications*, pages 136–154, 2012.
- [14] A. Inan, M. Kantarcioglu, and E. Bertino. Using anonymized data for classification. In *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, pages 429–440, March 2009.
- [15] D. Jurafsky and J. H. Martin. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition, 2008.
- [16] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.
- [17] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: multilevel traffic classification in the dark. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 229–240, 2005.
- [18] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos. Profiling the end host. In *Passive and Active Network Measurement, 8th International Conference, PAM 2007*, pages 186–196, 2007.
- [19] T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. *IEEE Trans. Dependable Secur. Comput.*, 2(2):93–108, 2005.
- [20] K.-F. Lee and H.-W. Hon. Large-vocabulary speaker-independent continuous speech recognition using hmm. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 123–126 vol.1, apr 1988.
- [21] M. Liberatore and B. Levine. Inferring the source of encrypted HTTP connections. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 255–263, 2006.
- [22] M. Ehrenfreund. NSA tracking phone locations on planetary scale. *The Washington Post*, 5 December 2013. Available: [http://www.washingtonpost.com/world/national-security/nsa-tracking-phone-locations-on-planetary-scale/2013/12/05/dfe21740-5db2-11e3-bc56-c6ca94801fac\\_story.html](http://www.washingtonpost.com/world/national-security/nsa-tracking-phone-locations-on-planetary-scale/2013/12/05/dfe21740-5db2-11e3-bc56-c6ca94801fac_story.html) [Last accessed: 6 December 2013], 2013.
- [23] J. McHugh, R. McLeod, and V. Nagaonkar. Passive network forensics: Behavioural classification of network hosts based on connection patterns. *SIGOPS Oper. Syst. Rev.*, 42(3):99–111, Apr. 2008.
- [24] N. Melnikov and J. Schönwälder. Cybermetrics: user identification through network flow analysis. In *Proceedings of the 4th international conference on Autonomous infrastructure, management and security*, pages 167–170, Zurich, Switzerland, 2010.
- [25] T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander. Timely and continuous machine-learning-based classification for interactive ip traffic. *IEEE/ACM Trans. Netw.*, 20(6):1880–1894, Dec. 2012.
- [26] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 User Fingerprinting. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 99–110, New York, NY, USA, 2007. ACM.
- [27] D. Plonka. Flowscan: A network traffic flow reporting and visualization tool. In *Proceedings of the 14th USENIX conference on System administration*, LISA '00, pages 305–318, Berkeley, CA, USA, 2000. USENIX Association.
- [28] R. Brandom. The NSA tracks the location of hundreds of millions of phones. *The Verge*, 4 December 2013. Available: <http://www.theverge.com/2013/12/4/5175778/the-nsa-tracks-the-location-of-hundreds-of-millions-of-phones> [Last accessed: 6 December 2013], 2013.
- [29] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [30] D. Riboni, A. Villani, D. Vitali, C. Bettini, and L. Mancini. Obfuscation of sensitive data for incremental release of network flows. *Networking, IEEE/ACM Transactions on*, PP(99):1–1, 2014.
- [31] D. Rossi and S. Valenti. Fine-grained traffic classification with Netflow data. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, pages 479–483. ACM, 2010.
- [32] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic. Who do you sync you are?: smartphone fingerprinting via application behaviour. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 7–12, 2013.
- [33] Q. Sun, D. Simon, and Y. Wang. Statistical identification of encrypted web browsing traffic. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 19–30, 2002.
- [34] I. Trestian, S. Ranjan, A. Kuzmanovi, and A. Nucci. Unconstrained endpoint profiling (googling the internet). In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication - SIGCOMM '08*, page 279, New York, New York, USA, 2008. ACM Press.
- [35] S. Wei, J. Mirkovic, and E. Kissel. Profiling and clustering internet hosts. In *Proceedings of the 2006 International Conference on Data Mining, DMIN 2006*, pages 269–275, Las Vegas, 2006.
- [36] C. V. Wright, L. Ballard, S. E. Coull, F. Monroe, and G. M. Masson. Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 35–49, Washington, DC, USA, May 2008. IEEE Computer Society.
- [37] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 169–180, 2005.
- [38] K. Xu, Z.-l. Zhang, and S. Bhattacharyya. Internet Traffic Behavior Profiling for Network. *IEEE/ACM Trans. Netw.*, 16(6):1241–1252, 2008.
- [39] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen. How much can behavioral targeting help online advertising? In *Proceedings of the 18th International Conference on World Wide Web, WWW '09*, pages 261–270, New York, NY, USA, 2009. ACM.
- [40] F. Zhang, W. He, X. Liu, and P. G. Bridges. Inferring users' online activities through traffic analysis. In *Proceedings of the Fourth ACM Conference on Wireless Network Security, WiSec '11*, pages 59–70, New York, NY, USA, 2011. ACM.
- [41] Z.-H. Zhou. *Ensemble Methods : Foundations and Algorithms*. CRC Press. 2012.