# A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures

EVA PAPADOGIANNAKI, Foundation for Research and Technology – Hellas
SOTIRIS IOANNIDIS, Technical University of Crete and Foundation for Research and Technology – Hellas

The adoption of network traffic encryption is continually growing. Popular applications use encryption protocols to secure communications and protect the privacy of users. In addition, a large portion of malware is spread through the network traffic taking advantage of encryption protocols to hide its presence and activity. Entering into the era of completely encrypted communications over the Internet, we must rapidly start reviewing the state-of-the-art in the wide domain of network traffic analysis and inspection, to conclude if traditional traffic processing systems will be able to seamlessly adapt to the upcoming full adoption of network encryption. In this survey, we examine the literature that deals with network traffic analysis and inspection after the ascent of encryption in communication channels. We notice that the research community has already started proposing solutions on how to perform inspection even when the network traffic is encrypted and we demonstrate and review these works. In addition, we present the techniques and methods that these works use and their limitations. Finally, we examine the countermeasures that have been proposed in the literature in order to circumvent traffic analysis techniques that aim to harm user privacy.



CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Networks** → **Network measurement**; **Network security**; **Mobile networks**; **Security protocols**; **Network privacy and anonymity**; • **Security and privacy** → **Intrusion detection systems**; **Mobile and wireless security**; **Security protocols**; **Web application security**

Additional Key Words and Phrases: Encrypted network traffic, encrypted network traffic analysis, network traffic inspection, network traffic processing, network analytics, application analytics, application usage analytics, QoSs analytics, QoE analytics, network security, network intrusion detection, mobile malware, user privacy, website fingerprinting, pii leakage, device fingerprinting, location estimation, network middlebox, network function, machine learning, deep learning, neural networks, searchable encryption, network traffic interception, network packet metadata

This work was supported by the projects CONCORDIA, CyberSANE, C4IIoT and COLLABS funded by the European Commission under Grant Agreements No. 830927, No. 833683, No. 833828, and No. 871518. This publication reects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Authors' addresses: E. Papadogiannaki, Foundation for Research and Technology - Hellas, Institute of Computer Science, N. Plastira 100, Vassilika Vouton, GR-70013 Heraklion, Crete, Greece; email: epapado@ics.forth.gr; S. Ioannidis, School of Electrical and Computer Engineering, Technical University of Crete Akrotiri Campus 73 100 Chania, Crete, Greece & Foundation for Research and Technology - Hellas, Institute of Computer Science, N. Plastira 100, Vassilika Vouton, GR-70013 Heraklion, Crete, Greece; email: sotiris@ece.tuc.gr.



0360-0300/2021/07-ART123 $15.00
https://doi.org/10.1145/3457904

**ACM Reference format:**
Eva Papadogiannaki and Sotiris Ioannidis. 2021. A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures. *ACM Comput. Surv.* 54, 6, Article 123 (July 2021), 35 pages.
https://doi.org/10.1145/3457904

## 1 INTRODUCTION

The adoption of network encryption is rapidly growing. The 2019 Annual Report of *Let's Encrypt* [19] states that in just 4 years, global HTTPS page loads have increased from 39% to more than 80% [16]. In 2019, 1 year after TLS 1.3 had been published as an RFC [15], IETF reports that its adoption is rapidly growing with 30% of Chrome's Internet connections to negotiate TLS 1.3 [18]. Even though network encryption is crucial for the protection of users and their privacy, it naturally introduces challenges for tools and mechanisms that perform deep packet inspection and rely heavily on the processing of packet payloads. Typical applications of deep packet inspection are packet forwarding and l7 filtering [6, 124], while it is a vital operation in firewalls, intrusion detection, and prevention systems [21–23]. In addition, the majority of content and service providers perform network analytics using **deep packet inspection** (**DPI**) to improve network performance and provide good quality of service and experience to their users. However, with the widespread adoption of network encryption protocols, solutions that rely on retrieving meaningful information from packet payload contents are becoming less and less effective and new mechanisms must be employed to keep up with network encryption.

In this survey, we present the works that we find in the literature that are able to perform traffic processing and inspection even when the network is encrypted. We examine the use cases of these works (e.g., network analytics) and how authors achieve the implementation of such systems. Having no visibility over the packet payload contents introduces major challenges. Thus, the goal of this survey is to identify the means to achieve encrypted network traffic analysis and inspection effectively. This survey will help researchers in the field to (i) understand the challenges of traffic inspection when the network traffic is encrypted or tunnelled, (ii) discover the use cases and applications of encrypted traffic analysis, (iii) acquire knowledge of the methods that are used to achieve encrypted traffic analysis, (iv) deduce which techniques are appropriate with respect to the objectives of a system, (v) recognize the constraints each method presents, and finally, (vi) identify the publicly available datasets that are appropriate for use.

Figure 1 displays the taxonomy that we propose for the works that we examine in this survey. First, we divide the works based on their use case and application goal. More specifically, we divide the works into four application domains: (i) the network analytics domain, (ii) the network security domain, (iii) the user privacy domain, and (iv) the domain of network functions in middleboxes. Each work can then be characterized by the technique that is used (i.e., manipulation of traffic metadata and characteristics, interception of encrypted traffic, and utilization of cryptographic functions) and its main objectives (i.e., functionality, programmability, and deployment).

In Section 3, we discuss the works that target the network analytics domain. We divide the works into more detailed categories and we dedicate one subsection to one sub-category (e.g., in Section 3.1 we present works that focus on application and protocol classification; in Section 3.2 we discuss works that identify application usage actions). Then, in Section 3.4 we discuss the algorithms and techniques used, in Section 3.5 we present the publicly available datasets used, and in Section 3.6 we examine the objectives and limitations of the works reviewed in the section. We follow the same paragraph organization format for each one of the Sections 4–6. For completeness, the rest of this article is organized as follows. Section 2 offers background information about network traffic processing and inspection that could potentially help the reader smoothly ingest

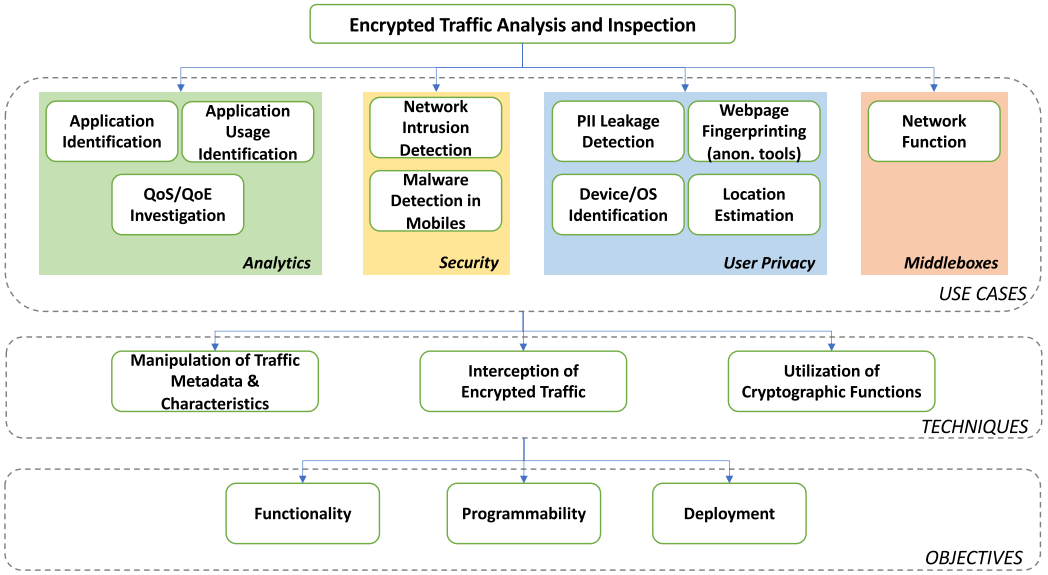


Fig. 1. A taxonomy for encrypted network traffic inspection works categorized by use case, technique and objective.

the remaining sections. In Section 7, we discuss the countermeasures that have been developed against encrypted network traffic analysis, and finally in Section 8, we discuss the lessons that we acquire from this survey. In the following paragraphs of this section, we present the related surveys that exist in the literature.

### 1.1 Related Surveys

In this section, we present the related works that can be found in the literature. More specifically, we research the most recent and popular surveys that exist in the domain of network traffic classification with a focus on works that target current trends in the area, such as encrypted and mobile networks. In 2015, Velan et al. [156] surveyed methods for encrypted traffic analysis and classified the literature using a multi-level taxonomy proposed in [81]. The taxonomy distinguishes the different traffic classification methods based on (i) the input (e.g., traffic payload or traffic properties), (ii) the technique (e.g., payload inspection, statistical or machine learning), (iii) the output type and (i,v) the dataset used. The survey of Velan et al. covers the works that aim for application protocol and type detection in encrypted streams. Our survey is more extensive; application and protocol identification in encrypted traffic is just a subset of the works that we survey. In 2018, Conti et al. [48] published a study for network traffic analysis specifically for mobile devices. They categorized the surveyed works based on the goal of the analysis (e.g., application identification, user fingerprinting), the point of traffic capturing (e.g., mobile devices, Wi-Fi access points), and the targeted mobile platform (e.g., Android). Their study includes works that inspect either unencrypted or encrypted traffic, but specifically on mobile devices. In this survey, we do not focus only on mobile devices. In 2019, Wang et al. [157] published a survey of deep learning techniques for traffic classification in mobile and encrypted networks. Similarly, Aceto et al. [28, 29] presented a review for mobile encrypted traffic classification works that use deep learning techniques. Kwon et al. [86] published a survey for deep learning–based network anomaly detection, which corresponds to a subcategory of the works that we survey. In 2019, Mohammed et al. [101] surveyed the

Table 1. A Comparison of this Survey Against Other Related Surveys

| | Domains examined | | | | Techniques surveyed | | | |
|---|---|---|---|---|---|---|---|---|
| Survey | Analytics | Security | Privacy | Mid/boxes | ML | Interception | Crypt. Fun. | Details |
| Velan et al. [156] | ✓ | – | – | ✓ | – | – | – | – |
| Conti et al. [48] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | mobile |
| Wang et al. [157] | ✓ | – | – | – | ✓ | – | – | mobile |
| Kwon et al. [86] | – | ✓ | – | – | ✓ | – | – | – |
| Mohammed et al. [101] | ✓ | – | – | – | ✓ | – | – | SDN |
| Sultana et al. [142] | – | ✓ | – | – | ✓ | – | – | SDN |
| Zhang et al. [173] | ✓ | ✓ | ✓ | ✓ | ✓ | – | – | mobile |
| Tahaei et al. [143] | ✓ | ✓ | ✓ | – | ✓ | – | – | IoT |
| Salman et al. [127] | ✓ | ✓ | ✓ | – | ✓ | – | – | – |
| Aceto et al. [28, 29] | ✓ | ✓ | – | – | ✓ | – | – | mobile |
| **This survey** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | counter/res |

We compare the surveys based on their coverage in (i) applications and use cases and (ii) techniques and methods.

techniques for traffic classification and prediction using machine and deep learning in **Software Defined Networks** (**SDNs**). Also in 2019, Sultana et al. [142] surveyed the SDN-based network intrusion detection systems that use machine learning algorithms. In 2020, Tahaei et al. [143] presented a survey regarding traffic classification in **Internet of Things** (**IoT**) networks. In 2020, Salman et al. [127] published a review on machine learning techniques for internet traffic classification, based on the different objectives for traffic classification, such as protocol classification, application classification, action classification, category classification, and device classification. Finally, Zhang et al. [173] presented how deep learning is used in wireless and mobile networking in an extensive survey. In Table 1, we present a comparison of this survey against other related surveys. In this survey, we present works in a wide range of encrypted network traffic classification and analysis applications, without targeting only one domain like related surveys (e.g., network security or analytics). In addition, we present the techniques and methods that have been proposed in the literature. Thus, we do not focus on surveying works that focus only on machine learning, for instance. Finally, along with the different encrypted network traffic analysis applications and techniques, we also present the countermeasures against traffic analysis that have been proposed in the literature.

## 2 NETWORK TRAFFIC PROCESSING AND INSPECTION

In this section, we present typical network traffic processing and inspection methods and techniques that have been widely used by the literature through the years. This section will help the reader comprehend how network traffic processing is achieved and identify the challenges that network encryption has introduced.

Anomaly detection is a technique applicable in a variety of domains, such as network intrusion detection, aiming to identify unusual or suspicious activity and behavior. An anomaly detection-based approach estimates a model of "normal network activity," and future activity on the network is evaluated with respect to its probability under the learnt model. This way, it is possible to distinguish a typical and potentially malicious network activity from benign. This classification is based on certain heuristics or rules and aims to detect misuses or abnormal behavior. Specifically, for an anomaly-based intrusion detection system to positively identify suspicious traffic, the system must be taught to recognize normal system activity [44]. For example, a number of volume-based detection techniques exist. These techniques monitor the traffic load of a network, to identify anomalies that trigger significant traffic volume changes (e.g., flooding attacks). Feature-based anomaly detection aims to fill the gap of volume-based systems by examining a range of network traffic features [83]. The two phases of the majority of anomaly detection systems consist of the training

phase (where a profile of normal behavior is constructed) and testing phase (current traffic is compared with the model created in the training phase). Anomalies are detected mostly using machine or deep learning techniques. However, anomaly detection often suffers from (i) high false-positive rates, (ii) the difficulty to obtain reliable training data, and (iii) the longevity of training data and behavioral dynamics of the system.

Shone et al. [135] conclude that the dynamic nature of modern networks, such as the ever increasing traffic speed, volume, and diversity, introduces an inevitable deficiency in creating a holistic and widely established anomaly detection solution. Indeed, the number of different protocols and the diversity of data in todays' connections introduce increased difficulty and complexity levels when attempting to differentiate between normal and abnormal behavior, increasing the difficulty in establishing an accurate norm and widens the scope for potential exploitation or zero-day attacks. Although anomaly-based detection techniques can be very effective in the domain of network security, signature-based techniques offer the ability to process and inspect a wider area of applications.

Signature-based network inspection is another technique for traffic classification and characterization and is mostly performed using DPI techniques. DPI is the core component for many systems that are part of the network, such as traffic monitors, classifiers, packet filters, network intrusion detection, and prevention systems. Network components use DPI in different layers of the OSI model. Unlike the early years of packet inspection, when DPI was applied in packet headers only (e.g., proxies and firewalls), nowadays, protocol complexity and obfuscation has led us to inspect the packet content in all encapsulated layers. In fact, governments, **Internet and Communications Service Providers** (**ISPs**/**CSPs**), among other organizations, heavily rely on DPI systems for accurate traffic monitoring and characterization. The use of DPI can result in better quality of service (QoS) by identifying different styles of content and streaming them accordingly. Similarly, by examining the contents of the incoming packets, suspicious and malicious traffic can be identified and examined. In the domain of network security, signature-based network inspection techniques can be very effective when similar attacks are known to security software developers *a priori*.

As already mentioned, DPI is the core procedure of typical applications in the domain of network security, analytics, and others. The requirements for complex traffic inspection, as well as the constant increase in network speeds, have motivated numerous works for constantly proposing novel DPI approaches. Network intrusion detection systems have become very powerful tools in the hands of network administrators and security experts over past decades, assisting in the detection and prevention of a wide range of attacks. Popular **network intrusion detection system** (**NIDS**) solutions like Snort [22] and Suricata [21] utilize pattern matching and regular expressions in order to analyze network traffic while Zeek/Bro [23] utilizes scripts that allow easier automation. The research community has also put effort into improving the performance of NIDS using either commodity hardware, such as **graphics processing units** (**GPUs**) [69, 70, 76, 116, 138, 152–155] and parallel nodes [117, 149], or specialized hardware, such as ternary content-addressable memories (TCAMs), application-specific integrated circuits (ASICs), and field-programmable gate array (FPGAs) [47, 99, 141, 169]. However, these works are able to process network traffic that is unencrypted, since they extract meaningful information from network packet payload content.

Traffic classification and network analytics is an important QoS component at central network traffic ingress points, as well as at end-host computers. In order to identify the incoming traffic accurately, a fine-grained analysis should be performed, inspecting both packet header and payload. The majority of traffic classification approaches are flow-based techniques that aim to characterize flows instead of packets, according to the application that creates them. Many works have proposed a number of methods to identify the application associated with a traffic flow [79, 103].

Table 2. Brief Presentation of Works in the Network Analytics Domain (Sorted by Category and Publication Year)

| Traffic Analysis Domain | Work | Category | Goal | Year of publication |
|---|---|---|---|---|
| Network Analytics | Karagiannis et al. [80] | Protocol/Application Identification | Network traffic classification using traffic behavioral patterns | 2005 |
| | Bernaille et al. [36] | | Application detection in SSL encrypted network connections | 2007 |
| | Wang et al. [158] | | iOS application classification | 2015 |
| | Alan et al. [31] | | Android application classification | 2016 |
| | Taylor et al. [146] | | Android application classification | 2016 |
| | Lopez et al. [94] | | IoT traffic classification | 2017 |
| | Taylor et al. [147] | | Android application classification | 2018 |
| | Aceto et al. [26] | | Mobile application classification | 2018 |
| | Aceto et al. [27] | | Mobile application classification | 2019 |
| | Aiolli et al. [30] | | Identification of user activities on smartphone-based Bitcoin wallet apps | 2019 |
| | Yao et al. [168] | | IoT traffic classification | 2019 |
| | Ede et al. [151] | | Mobile application classification | 2020 |
| | Coull et al. [54] | Application Usage Identification | Identification of user actions in iMessage | 2014 |
| | Conti et al. [49, 50] | | Identification of user actions on Android devices | 2016 |
| | Saltaformaggio et al. [128] | | Identification of user actions on mobile devices | 2016 |
| | Fu et al. [68] | | Identification of user actions on mobile messaging applications | 2016 |
| | Liu et al. [93] | | Identification of user actions on mobile messaging applications | 2017 |
| | Papadogiannaki et al. [114] | | Identification of user actions on mobile Over-The-Top applications | 2018 |
| | Wang et al. [162] | | Identification of user actions on mobile payment applications | 2019 |
| | Jiang et al. [77] | | Identification of user actions in remote desktop traffic | 2019 |
| | Wright et al. [165] | Stream Decoding | VoIP conversation decoding | 2008 |
| | Schuster et al. [129] | | Video stream decoding | 2017 |
| | Dimopoulos et al. [58] | QoS/QoE Investigation | Detection of QoE degradation | 2016 |
| | Orsolic et al. [110] | | Estimation of QoE in YouTube | 2016 |
| | Mazhar et al. [98] | | Investigation of video QoS in HTTPS and QUIC protocols | 2018 |
| | Khokhar et al. [82] | | Estimation of QoE in YouTube | 2019 |
| | Xu et al. [167] | | Investigation of mobile ABR video adaptation behavior under HTTPS and QUIC | 2020 |

A simple approach is to examine TCP port numbers, since well-known applications have specific port numbers (e.g., HTTP uses port 80, SSH uses port 22). Yet, aiming to hide from firewalls and network security tools, many applications and tools use dynamic port negotiation; of course, this makes port-based traffic classification weak. Other approaches propose the deep packet inspection against packet payload contents. Of course, this technique expects unencrypted packet payloads; however, assuming unencrypted packet payloads is not a realistic scenario in the present. The research community has investigated other ways to extract meaningful, side-channel information (packet- or network-level) that is able to be expressed into signatures. We discuss the methods of these works in the following sections.

## 3 ANALYTICS AFTER NETWORK ENCRYPTION

In this section, we discuss the literature for network analytics in encrypted communications. More specifically, we present works that focus on protocol and application classification (Section 3.1), application usage identification (Section 3.2), as well as the investigation of quality and experience in encrypted networks (Section 3.3). The works that are presented in this section are summarized in Table 2.

### 3.1 Protocol/Application Identification

There is a large number of works that examine the feasibility of traffic classification even when the communications are encrypted. The majority of such works focus on classifying the traffic's nature (e.g., video streaming, p2p traffic) and automating the classification procedure.

BLINC [80] was one of the pioneer papers that aimed to classify the network traffic in the dark, having no access to packet payloads, no knowledge of port numbers; only information that flow collectors provide, solely based on the host behavioral patterns. Bernaille et al. [36] propose a method to detect different applications even in encrypted communication channels, by observing the first packets of an SSL connection and their sizes. This enables them to recognize the application soon enough, achieving an 85% accuracy and early classification.

Wang et al. [158], taking advantage of the fact that mobile applications produce more identifiable traffic patterns, perform a packet-level analysis to determine what applications a single individual is using, exploiting side-channel information (e.g., traffic bursts) that is exposed inside the network traffic that is generated by mobile devices. To perform mobile app classification, the authors use random forests for 13 selected iOS applications achieving a classification accuracy of more than 68.59% with selected features and more than 87.23% with complete features.

Alan et al. [31] investigate whether Android applications can be identified through their network traffic launch-time using only the contents of TCP/IP headers. The experiments were conducted using 1,595 applications on four distinct Android devices. The authors made use of supervised learning methods to identify the apps that generated the traffic. Their approach is based on packet sizes, observed within the launch time traffic, since they are expected to yield good feature sets for application identification.

AppScanner [146] automatically fingerprints Android applications even in encrypted traffic. For the generation of the fingerprints, authors collect network traffic traces on the mobile device while running the corresponding applications. The application classification is conducted using a supervised learning algorithm that is fed with features that are exported through the collection of network traces. This scalable framework implementation is able to identify the profiled applications (110 most popular applications in Google Play Store) with more than 99% accuracy.

Stringoid [120] is a static analysis tool that estimates constructed URL strings using string concatenations in Android applications. The purpose of this work is to analyze web requests that originate from Android mobile applications. The authors use a dataset of 20 randomly selected Android applications, and with the Stringoid tool, they extract URLs from 30,000 Android applications.

Lopez-Martin et al. [94] propose using a **recurrent neural network** (**RNN**) combined with a **convolutional neural network** (**CNN**) for IoT traffic classification. The advantage of this work is that it outperforms alternative algorithms for traffic classification, while it does not require any feature engineering when applying new models.

In a succeeding work, Taylor et al. [147] show that a passive eavesdropper is able to identify mobile applications by fingerprinting the network traffic that they send, despite encryption. Again, using AppScanner and machine learning techniques they exploit the information that lays in network traffic, such as packet size and direction. In addition, they investigate how application fingerprints change over parameters like time, diversity of devices, and versions.

Aceto et al. [26] aim to improve the classification performance of mobile applications by proposing a Multi-Classification system, which combines specific decisions from base classifiers explicitly devised for mobile and encrypted traffic classification. The dataset that the authors used for testing, was collected by a mobile solutions provider. In a succeeding work, Aceto et al. [27] perform mobile traffic classification in encrypted network flow using deep learning techniques.

Shen et al. [133] perform encrypted traffic classification of decentralized applications on Ethereum using a feature selection of packet lengths, bursts, and time series. Aiolli et al. [30] identify user activities on Bitcoin wallet applications in mobile devices and are commonly used for sending, receiving, and trading Bitcoin.

Yao et al. [168] perform IoT traffic classification for smart cities using a method that relies on a deep learning aided capsule network for efficient classification. Their proposed work eliminates the process of manually selecting traffic features. FLOWPRINT [151] offers mobile application identification by analyzing the network traffic. It introduces an approach for application fingerprinting by combining destination-based clustering, browser isolation, and pattern recognition (in a semi-supervised manner). It is able to construct mobile application fingerprints for not known applications. Authors evaluate FLOWPRINT and they find that it is able to perform an accuracy of 89.2%. Even after application updates or newly encountered applications, FLOWPRINT has a precision of 93.5%. As ground-truth, authors use publicly available datasets. As features, authors extract all header values controlled by the communicating app as well as the sizes and inter-arrival times of packets. In addition, for the size and time-related features authors compute statistical properties, such as the maximum, standard deviation, and mean absolute deviation values.

### 3.2 Application Usage Identification

The works presented in this section offer fine-grained application event identification over encrypted traffic, often with the use of machine learning techniques.

Coull and Dyer [54] propose a method for traffic analysis of encrypted messaging services. More specifically, the authors aim to show that an eavesdropper would be able to retrieve fine-grained information by the communication channel, such as specific user actions, the size of messages that are exchanged, or even the language that is being used for the communication. Their results demonstrate the feasibility of gaining information by observing packet lengths, but their analysis is limited to Apple's iMessage application and is an offline study.

Conti et al. [49, 50] propose a system to analyze encrypted network traffic to identify user actions on Android devices, such as email exchange, interactions over social network, and so on. Their framework uses TCP/IP packet fields, like IP addresses and ports, along with other features, like packet size, direction, and timing. They analyze numerous Android applications with diverse functionalities, such as Gmail, Facebook, Twitter, Tumblr and Dropbox. Using machine learning, they achieve high accuracy and precision for the identification of different user actions in each tested Android application (e.g., mail exchange, posting a photo online, or publishing a tweet).

NetScope [128] is a work that performs robust inference of users' activities, for both Android and iOS devices, based on inspecting IP headers. This work demonstrates how a passive eavesdropper is capable of identifying fine-grained user activities within a network (even over encrypted communication channels) generated by the applications used. Based on the intuition that the implementation of each individual mobile application leaves a fingerprint on its traffic behavior, such as transfer rates and packet exchanges, NetScope learns the subtle traffic behavioral differences between user activities becoming able to distinguish them.

Fu et al. [68] propose an approach to classify usage in mobile messaging applications. Their system, namely, CUMMA, classifies the usage in mobile messaging applications by taking into account user behavioral patterns, network traffic characteristics, and temporal dependencies. More specifically, they show that the observation of packet lengths and time delays can allow the classification of WhatsApp and WeChat traffic and identify the corresponding usage types (e.g., photo sharing). With this framework, the authors achieve 96% and 97% accuracy in WeChat and WhatsApp, respectively.

Liu et al. [93] develop an analyzer to classify encrypted mobile traffic to application usage activity. Using similarity measurements, the authors select discriminative features from traffic packet sequences. For their online analyzer, the authors represent a traffic flow with a series of time windows. For their experiments, they analyze WeChat, WhatsApp, and Facebook applications.

OTTer [114] is a scalable engine that identifies fine-grained user actions, like voice/video calls or messaging, in popular Over-The-Top mobile applications, such as WhatsApp, Skype, Viber, and Facebook Messenger in encrypted network traffic connections. The engine operates at traffic loads with an average of 109 Gbps.

Wang et al. [162] identify financial transactions at the trading stage via analyzing the encrypted network traffic, by identifying the mobile payment app from traffic data, classifying specific actions on the mobile payment app, and finally, detecting the detailed steps within the action.

Jiang et al. [77] investigate if remote desktop traffic, even if encrypted, can reveal usage information. Indeed, their results show the feasibility of this, taking advantage of side-channel information leakage.

*VoIP Conversation and Video Stream Decoding.* There are also works that use traffic analysis to extract voice information from encrypted VoIP conversations or identify encrypted video streams.

For example, Wright et al. [165] show that when the transmitted audio is encoded using variable bit rate codecs, the length of VoIP packets can be used to recognize words or phrases within a standard speech corpus. This means that a passive observer can identify phrases even in encrypted calls with an average accuracy of 50%. Schuster et al. [129] explain the root causes of burst patterns in encrypted video streams, show how to exploit these patterns for video identification, develop and evaluate a noise-tolerant identification methodology based on deep learning and, finally, they demonstrate how an attacker without direct observations of the network can identify videos being streamed. The features that authors use are flow attributes, such as down/up/all bytes per second, down/up/all packet per second, and down/up/all average packet length. The applications that authors examine are Netflix, Youtube, Amazon, and Vimeo.

### 3.3 Quality of Service/Experience Investigation

Streaming video content on mobile devices is a trend that is continually growing among users. This causes a tremendous demand for higher bandwidth and better provisioning throughout the network infrastructure. End-to-end encryption, though, leaves providers with limited indicators for identifying QoE issues. Thus, the works presented in this section aim to measure QoS and QoE from the perspective of a telecommunication service provider that has only visibility on the network traffic that is often encrypted.

Dimopoulos et al. [58] propose models able to detect different levels of QoE degradation that is caused by stalling, average video quality, and quality variations. The predictive models that the authors develop are evaluated on the production network of a large-scale mobile operator, where the authors show that their system is able to accurately detect **Quality of Experience** (**QoE**) problems with up to 92% accuracy. The significant features that the authors extract are **Round-trip time (RTT)**-related, bytes transmitted, packet loss percentage, and other network-related features.

Orsolic et al. [110] use machine learning for the estimation of YouTube Quality of Experience. To test their approach, the authors collect more than 1k different YouTube video traces under different bandwidth scenarios. Mazhar et al. [98] investigate the QoS of video in HTTPS and QUIC protocols. The set of features that expose usable information is based on (i) network and transport layer header information for TCP flows, and (ii) network layer features (based on inter-arrival time, packet sizes, packet/byte counts, throughput) for QUIC flows. Khokhar et al. [82] put YouTube under experimentation and perform network traffic measurements for QoE estimation using

Table 3. Techniques and Algorithms Used by Works in the Network Analytics Domain (Sorted by Category and Publication Year)

| Work | Category | Algorithm/Technique | Performance Evaluation Metrics |
|---|---|---|---|
| Karagiannis et al. [80] | Protocol/Application Identification | Graphs, Statistics | Completeness, Accuracy |
| Bernaille and Teixeira [36] | | Clustering with Gaussian Mixture Model | True-/False-Positive Rates |
| Wang et al. [158] | | Classification with **Random Forests (RFs)** | Estimated accuracy, Overall accuracy, True-Positive Rate |
| Alan and Kaur [31] | | Classification with Jaccard's coefficient, Gaussian Naïve Bayes, and Multinomial Naïve Bayes | Accuracy |
| Taylor et al. [146] | | Classification with Multi-class **Support Vector Machine (SVM)**, Multi-class RF, Binary SVM, and Binary RF | Speed of training, Size of classifier, Confidence per classification, True Negatives, Robustness |
| Lopez-Martin et al. [94] | | Classification with Recurrent Neural Network combined with a Convolutional Neural Network (CNN) | Accuracy, F1-score, Precision, Recall |
| Taylor et al. [147] | | Classification with Multi-Class Support Vector Machine (SVM), Multi-Class RF, Binary SVM, and Binary RF | Precision, Recall, F1-score, Accuracy |
| Aceto et al. [26] | | Classification with Naïve Bayes, Multinomial Naïve Bayes, Random Forests, Support Vector, Decision Trees | Accuracy, Precision, Recall, F-measure |
| Aceto et al. [27] | | Classification with Convolutional Neural Network | Accuracy, F-measure, **Runtime Per-Epoch (RTPE)** |
| Aiolli et al. [30] | | Classification with RFs and SVM | Precision, Recall, F1-score |
| Yao et al. [168] | | CNN, Convolutional Capsule Network, Fully-connected Capsule Network, **Long Short-Term Memory (LSTM)** | Accuracy, F1-score, precision, recall |
| Ede et al. [151] | | Semi-supervised fingerprinting with Clustering and cluster correlation | F1-score, Precision, Recall, Accuracy, Robustness |
| Coull and Dyer [54] | Application Usage Identification | Classification with Binomial Naïve Bayes | Accuracy |
| Conti et al. [49, 50] | | Classification with RFs | F-measure, Accuracy, Precision, Recall |
| Saltaformaggio et al. [128] | | Classification with SVM and Clustering with $k$-means | Detection time and True Positives, Misclassifications, False Negatives, False Positives, Precision, Recall |
| Fu et al. [68] | | Classification with Gradient Boosted Trees, SVM, Naive Bayes, KNeighbors | Accuracy, Precision, Recall, F-measure |
| Liu et al. [93] | | Classification with RFs and Clustering with recursive Constrained KMeans | Accuracy, Precision, Recall, F-measure, Processing Throughput (pps) |
| Papadogiannaki et al. [114] | | Frequent Pattern Mining for Signature Generation | True Positives, False Positives, and Performance Throughput |
| Wang et al. [162] | | Classification with RFs, Ada Boost, and **Gradient Boosting Decision Tree (GBDT)** | Accuracy, Recall, Precision, F1-score |
| Jiang et al. [77] | | Classification with **Logistic Regression** (LR), SVM, GBDT, and RFs | TPR, FPR, and F1-score |
| Wright et al. [165] | Stream Decoding | **Hidden Markov Models (HMMs)** | Recall, Precision |
| Schuster et al. [129] | | Gaussian distribution for Maximum Likelihood Estimator and CNN | Precision, Recall, Delay |
| Dimopoulos et al. [58] | QoS/QoE investigation | Classification with RFs | True Positives, False Positives, Precision, Recall |
| Orsolic et al. [110] | | Classification with RF, Naïve Bayes, SVM, Decision Trees | Accuracy |
| Mazhar and Shafiq [98] | | Classification with Decision Trees | Precision, Recall |
| Khokhar et al. [82] | | Classification with RF, Linear Regression, and RF Regression | Precision, Recall, F1-score, Accuracy, **Root Mean Square Error (RMSE)** |
| Xu et al. [167] | | Fingerprint construction from chunk sizes | Accuracy |

The table also includes the metrics that authors use to evaluate the work.

network-related features, as well. CSI [167] infers mobile ABR video adaptation behavior under HTTPS and QUIC using packet size and timing information.

### 3.4 Techniques

The majority of the works discussed in this section employ machine learning techniques to investigate the feasibility of traffic analysis and usage classification even when the communications are encrypted. In this section, we present the techniques and algorithms that are more popular among the works of this category. A detailed overview of the techniques used for classification can be found in Table 3. As illustrated in Table 3, machine and deep learning techniques are very popular in the domain of network analytics. More specifically, it seems that supervised or semi-supervised algorithms are mostly used for traffic classification and network analytics. These algorithms perform well with labeled and big datasets, while they require training. For instance, for mobile application classification, the authors choose Multinomial Naive Bayes (e.g., [26]), Support

Table 4. Datasets Used by Works in the Network Analytics Domain
(Sorted by Category and Publication Year)

| Work | Category | DatasetAvailability | Dataset Details |
|---|---|---|---|
| Yao et al. [168] | Protocol/Application Identification | Public | USTC-TFC2016 Dataset [12] |
| Ede et al. [151] | | Public | Recon Dataset [14] |

Vector Machine (e.g., [146]), and Hidden Markov Models (e.g., [165]) algorithms as well as other classifiers, such as Random Forest, Decision Trees, Gaussian Naive Bayes [31], and the $k$-Nearest Neighbors algorithm for pattern recognition [60]. For a more detailed classification (i.e, the identification of user actions and events inside mobile applications), the authors choose hierarchical clustering techniques [50, 68].

Besides machine learning, it seems that recently, researchers have turned to neural networks since they perform better than single machine learning algorithms. A neural network combines different machine learning algorithms for modeling data using graphs of neurons, while it is able to make accurate decisions and learn from its own errors. This makes a neural network work independently without requiring any human intervention.

Being encrypted, network packet payloads do not offer significant information. Thus, most techniques discussed in this section take advantage of data that are available in packet headers. The majority of these techniques use information like network packet sizes, directions, and time-related data, which are treated as features to train the corresponding machine learning models. More specifically, many works use as features the packet size and the packet direction [31, 40, 50, 54, 75]. Herrman et al. also use the IP packet length distribution [75]. Selecting a subset of packets in a single network flow is also common. For instance, Lu et al. do not consider incoming MTU packets [96], while Bernaille et al. keep only the first packets [37].

In Table 3, we also display the metrics used for the evaluation of each technique. Since the majority of these works are based on machine learning, they mostly measure their technique's (i) accuracy, (ii) precision, (iii) recall, (iv) true-, and (v) false-positive rates. However, even though the majority of the works discussed in this section use similar evaluation metrics, we would not be able to properly compare their effectiveness since they use different datasets to train their systems. Finally, we notice that only few works present processing evaluation metrics, such as memory consumption, training time, or throughput. In the next section (Section 3.5), we discuss the datasets used for the works of traffic classification for network analytics.

### 3.5 Datasets

Table 4 presents the public datasets that were used in the domain of network analytics. In the category of application identification and classification, Yao et al. [168] used the USTC-TFC2016 dataset [12] that among others contains network traffic from BitTorrent, Facetime, Gmail, and Skype. Ede et al. [151] used the Recon dataset [14], which consists of labeled network traces of 512 Android apps from the Google Play Store, including multiple versions for over a period of 8 years.

We notice that the vast majority of the works in this section do not use public datasets to train their models. The proprietary datasets used come either by real or emulated usage representing network usage with applications of interest, either in mobile or fixed networks. The absence of public datasets that contain network traffic that is both encrypted and labeled is apparent. Thus, authors are producing their own datasets.

### 3.6 Objectives and Limitations

The majority of the works in the category of network analytics focus on the functionality of their approaches. More specifically, authors focus on the thorough examination of the traffic analysis feasibility when the network traffic is encrypted. Indeed, they show that it is possible to detect the nature of the traffic in a fine-grained manner. For instance, Conti et al. [49] are able to identify different actions (e.g., post a tweet or send a message) in mobile applications (such as Twitter and Facebook) accurately even when the network traffic is encrypted. Yet, there are works that except for functionality, they aim for programmability and deployment as well [93, 114]. OTTer achieves a detailed characterization of usage (i.e., video call, voice call, chat) in different Over-The-Top applications like Skype and WhatsApp [114]. It is also integrated into a DPI engine that is deployed in a live traffic test-bed with an average of 109 Gbps.

All the works that use machine learning algorithms for the classification of network traffic need to be retrained in order to remain robust across new and diverse data. In addition, the majority of such works choose a subset of applications or protocols in order to examine the feasibility of classification. This makes such solutions mostly effective only for applications and protocols used for training, something that introduces potential scalability and adaptability issues. Even while scoring high performance (with metrics like accuracy, precision, and recall) in close-world scenarios (i.e., under a specific ground-truth dataset), these systems will certainly produce high rates of false positives when tested against real-world datasets of traffic traces. Furthermore, numerous countermeasures exist that can perform against traffic analysis techniques (such countermeasures are discussed in Section 7), making many of the works discussed in this section unable to bypass them.

## 4 SECURITY AFTER NETWORK ENCRYPTION

In this section, we present the state-of-the-art on encrypted network traffic analysis for network security. The works that are presented in this section are summarized in Table 5.

### 4.1 Intrusion Detection

Some techniques focus on identifying malicious behavior in the network, examining the characteristics of the underlying traffic, using exclusively machine learning approaches.

Taleb et al. [65, 144] propose an approach that identifies misuses in encrypted protocols with network packet inspection that focuses on processing of packet header information. Amoli et al. present a real-time unsupervised NIDS, able to detect new and complex attacks within encrypted and plaintext communications [32]. Anderson and McGrew [33] compare the properties of six different machine learning algorithms for encrypted malware traffic classification.

Shone et al. [135] propose a system that combines deep learning techniques for network intrusion detection. For the evaluation of their system, the authors use the KDD Cup '99 and NSL-KDD datasets with high accuracy (almost 90%). Kitsune [100] is a NIDS, based on neural networks, and designed for the detection of abnormal patterns in network traffic. It monitors the statistical patterns of recent network traffic and detects anomalous patterns.

Papadogiannaki et al. [113, 115] perform network intrusion detection by generating signatures from packet metadata sequences. They evaluate their methodology using the UNSW-NB15 dataset [104] that contains network traffic traces for numerous attacks.

Tang et al. [145] present a deep learning approach for flow-based anomaly detection in SDN environments. The authors build a **Deep Neural Network** (**DNN**) model for an intrusion detection system and train it with the NSL-KDD dataset, using six basic features of the NSL-KDD dataset. Niyaz et al. [109] utilize deep learning in order to detect DDoS attacks in SDN environments. The

Table 5. Works in the Security Domain, Sorted by Category and Publication Year

| Traffic Analysis Domain | Work | Category | Goal | Year of Publication |
|---|---|---|---|---|
| Network Security | Amoli et al. [32] | Network intrusion detection | Real-time network intrusion detection within encrypted communications | 2016 |
| | Anderson and McGrew [33] | | Encrypted malware traffic classification | 2017 |
| | Shone et al. [135] | | Network intrusion detection using a combination of DL techniques | 2018 |
| | Mirsky et al. [100] | | Neural network–based network intrusion detection system | 2018 |
| | Papadogiannaki et al. [113, 115] | | Intrusion detection with signatures from packet metadata sequences | 2020 |
| | Tang et al. [145] | Network intrusion detection (SDN) | Flow-based anomaly detection | 2016 |
| | Niyaz et al. [109] | | DDoS attack identification | 2016 |
| | Shabtai et al. [130] | Malware detection on mobile devices | Malware detection on Android mobile devices | 2012 |
| | Shabtai et al. [131] | | Identification of malicious attacks or masquerading/injected mobile applications | 2014 |
| | Wang et al. [159] | | Detection of mobile malware behavior using network traffic | 2016 |
| | Lashkari et al. [88] | | Detection of malicious or masquerading mobile applications | 2017 |

proposed system identifies individual DDoS attacks with an accuracy of almost 96% and classifies the traffic into benign or attack traffic, with an accuracy of 99.82% with low false positives.

### 4.2 Intrusion and Malware Detection in Mobile Devices

While the ever increasing adoption of traffic encryption has significantly improved the user privacy and security, traditional intrusion detection systems based on inspecting unencrypted traffic are becoming obsolete. Thus, there is a number of works that aim to detect malicious bahavior on mobile devices, mainly relying on machine learning approaches.

Andromaly [130] is a framework for malware detection on Android mobile devices. The host-based malware detection system monitors features and events that are retrieved from mobile devices and applies anomaly detection for the classification of the collected data.

Shabtai et al. [131] propose a system that identifies (i) attacks or masquerading applications installed on a mobile device and (ii) injected applications with malicious code.

In TrafficAV [159], the mobile network traffic is mirrored from the wireless access point to the server for data analysis. The data analysis and malware detection are performed on the server side. TrafficAV performs network traffic analysis in multiple levels. The proposed method combines network traffic analysis with a machine learning algorithm (i.e., C4.5 decision tree) and is able to identify malware in Android devices with good accuracy results. In an evaluation with 8,312 benign apps and 5,560 malware samples, the TCP flow detection model and the HTTP detection model achieve detection rates of up to 98% and 99.65%, respectively.

Lashkari et al. [88] detect malicious and masquerading applications on mobile devices. Their proposed method shows a good average accuracy (91.41%) and precision (91.24%) with a low false-positive rate. The authors use five different classifiers: Random Forest (RF), K-Nearest Neighbor (KNN), Decision Tree (DT), Random Tree (RT), and Regression (R). The authors have published a labeled dataset of mobile malware traffic that contains benign Android applications or injected applications (e.g., with adware or other types of malware).

### 4.3 Techniques

In this section, we present the techniques and algorithms that are more popular among the works of this category. Table 6 displays works that perform intrusion detection even in encrypted networks. The majority of these works utilize machine learning algorithms for intrusion detection and classification.

Table 6. Techniques and Algorithms Used by Works in the Network Security Domain (Sorted by Category and Publication Year)

| Work | Category | Algorithm/Technique | Performance Evaluation Metrics |
|---|---|---|---|
| Amoli et al. [32] | Network intrusion detection | DBSCAN-based outlier detection, $k$-means-based outlier detection | FPR, TPR, Accuracy, Precision, Recall |
| Anderson et al. [33] | | Classification with Linear Regression, Logistic Regression, Decision Tree, Random Forest, SVM, Multi-layer Perceptron | Accuracy, Classification Time |
| Shone et al. [135] | | Auto-encoder Deep Neural Network | Accuracy, Precision, Recall, F-measure, False alarm, Training/testing Time |
| Mirsky et al. [100] | | **Isolation Forests (IFs)** and Gaussian Mixture Models and Deep Neural Network Auto-encoders with Ensemble Layer and Output Layer | TPR, FNR, and Processing Throughput |
| Papadogiannaki et al. [113, 115] | | Signature generation for Intrusion Detection using packet metadata sequences | True Positives, False Positives, and Performance Throughput |
| Tang et al. [145] | Network intrusion detection (SDN) | Deep Neural Network with an input layer, three hidden layers, and an output layer | Accuracy, Precision, Recall, and F-measure |
| Niyaz et al. [109] | | Stacked Autoencoder Deep Neural Network | Precision, Recall, F-measure |
| Shabtai et al. [130, 131] | Malware detection on mobile devices | Linear Regression, Decision Table, Support Vector Machine for Regression, Gaussian Processes for Regression, Isotonic Regression, Decision/Regression tree (REPTree) | TPR, Detection time, FPR, False Alerts, Memory and CPU consumption |
| Wang et al. [159] | | Classification with C4.5 Decision Tree | TPR, FPR |
| Lashkari et al. [88] | | Classification with Random Forest, $k$-Nearest Neighbor, Decision Tree, Random Tree, Regression | Accuracy, Precision, FPR |

The table also includes the metrics that authors use to evaluate the work.

Besides machine learning, researchers make use of neural networks and deep learning techniques. For intrusion detection, Shone et al. [135] propose a deep learning classification model constructed using stacked **non-symmetric deep autoencoders** (**NDAEs**). Tang et al. [145] present a deep learning approach for flow-based anomaly detection in SDN environments. The authors build a DNN model for intrusion detection and train it with the NSL-KDD dataset, using basic features found in the dataset. Niyaz et al. [109] utilize deep learning in order to detect DDoS attacks in SDN environments. Kitsune [100] is a NIDS, based on neural networks, and designed for the detection of abnormal patterns in network traffic. It monitors the statistical patterns of recent network traffic and detects anomalous patterns. Again, the majority of these techniques take advantage of data that are available in packet headers like network packet sizes, directions, and time-related data. More sophisticated techniques make use of additional, more advanced features, like the TLS handshake information [33, 67].

In Table 6, we also display the metrics used for the evaluation of each technique. Similarly to Section 3, the most common evaluation metrics are (i) accuracy, (ii) precision, (iii) recall, (iv) true-, and (v) false-positive rates, while few of the works report classification time and processing throughput.

### 4.4 Datasets

Table 7 presents the public datasets that were used for intrusion detection. The DARPA dataset [1] is one of the most popular datasets to train and evaluate intrusion detection systems. The ISCX-IDS-2012 intrusion detection dataset [25] consists of 7 days of benign and malicious network activity. The KDD Cup '99 dataset [2] includes a wide variety of intrusions simulated in a military network environment. The NSL-KDD dataset [7] is suggested to solve some of the inherent problems of the KDD '99 dataset. Finally, Moustafa and Slay [104] build and make publicly available a handy dataset for network intrusion detection systems, namely, UNSW-NB15 [10]. This dataset is used for evaluation in [113, 115].

Table 7. Datasets Used by Works in the Network Security Domain (Sorted by Category and Publication Year)

| Work | Category | Dataset Availability | Dataset Details |
|---|---|---|---|
| Amoli et al. [32] | Network intrusion detection | Public | DARPA [1], ISCX-IDS-2012 [25] |
| Shone et al. [135] | | Public | KDD Cup '99 [2], NSL-KDD [7] |
| Papadogiannaki et al. [113, 115] | | Public | UNSW-NB15 [10] |
| Tang et al. [145] | | Public | NSL-KDD [7] |

As Anderson and McGrew [33] correctly point out, finding the most proper features for classification with high accuracy, recall, and precision, is not a trivial procedure, while it is highly dependable on the ground-truth dataset collection that is available each time. However, user privacy-related regulations make data sharing a very challenging practice for organizations and data holders. In Table 7, we can notice that the NSL-KDD dataset [7] is the most popular dataset that is publicly available. Yet, we observe that the majority of the datasets that are used for intrusion detection are not very recent. For instance, Amoli et al. [32] and Shone et al. [135] use the DARPA [1] and KDD Cup '99 [2] datasets, respectively, both of which were more than 15 years old at the time the papers were published.

### 4.5 Objectives and Limitations

As most works that perform encrypted network traffic inspection for intrusion detection use machine learning techniques to examine the feasibility of attack classification, the primary objective is functionality. They train their models offline providing poor support for online intrusion detection when it comes to encrypted communications. The works that also aim for programmability, except for functionality, are [100, 113, 115, 130, 131]. In Section 6, we discuss middleboxes that can be deployed for online traffic inspection. Some of the network functions of these middleboxes can be used for intrusion detection and firewall applications.

The majority of the works discussed in this section can be bypassed by traffic analysis resistant techniques. Anderson et al. [33] examine and present mistakes and limitations in the network traffic analysis literature, such as the utilization of old and unreliable ground-truth datasets. Indeed, if we examine the datasets that are used as ground-truth for training most of the machine learning models in this section, are not very recent.

## 5 USER PRIVACY AFTER NETWORK ENCRYPTION

Besides network analytics and security, traffic analysis has been also used to monitor and profile the characteristics of mobile applications. This kind of tools empower the user to gain (i) insight regarding the applications used and (ii) control over **personally identifiable information** (**PII**) handling. Furthermore, in this section we discuss works that are able to fingerprint websites and applications even when anonymity tools like TOR [24] are used to hide the activity's nature, the user's identity, or the user's location. Finally, we present works that enable OS and device identification even in encrypted networks. The works that are presented in this section are summarized in Table 8.

### 5.1 Endpoint Device Tools and PII Leakage Detection

ProfileDroid [163] is a monitoring and profiling system that can characterize the behavior of Android applications at the static, user, OS, and network layers. The authors evaluate multiple Android applications, which present privacy and security, operational and performance issues. The authors evaluate the proposed tool using free and paid Android applications, observing (i) discrepancies between the app specification and app execution, (ii) higher costs resulting from free versions of apps, due to an order of magnitude increase in traffic, (iii) a great amount in network

Table 8. Works in the User Privacy Domain, Sorted by Category and Publication Year

| Traffic Analysis Domain | Work | Category | Goal | Year of Publication |
|---|---|---|---|---|
| User Privacy | Herrmann et al. [75] | Webpage identification | Webpage identification through OpenSSH, OpenVPN, CiscoVPN, Stunnel, TOR, JonDonym | 2009 |
| | Lu et al. [96] | | Webpage identification through SSH and SSL tunnels | 2010 |
| | Panchenko et al. [111, 112] | | Webpage identification through TOR | 2011 |
| | Cai et al. [40] | | Webpage identification through TOR and HTTPOS | 2012 |
| | Kwon et al. [84] | | Webpage identification through TOR hidden services | 2015 |
| | Draper-Gil et al. [60] | | Detection and characterization of VPN traffic | 2016 |
| | Cruz et al. [55] | | Identification of BitTorrent traffic in SSH tunnels | 2017 |
| | Lotfollahi et al. [95] | | Application identification in VPN traffic | 2017 |
| | Shahbar and Zincir-Heywood [132] | | Identification of anonymity networks | 2017 |
| | Montieri et al. [102] | | Traffic classification of anonymity tools | 2019 |
| | Chen et al. [46] | Device/OS identification | OS identification, NAT and tethering detection | 2014 |
| | Sivanathan et al. [136] | | IoT device identification using traffic features | 2018 |
| | Skowron et al. [137] | | Investigation of fingerprinting attacks targeting IoT devices | 2020 |
| | Lastovicka et al. [89] | | OS identification using TLS fingerprints | 2020 |
| | Ateniese et al. [35] | Location estimation | Position extrapolation through location-based encrypted traffic | 2015 |
| | Razaghpanah et al. [121] | PII leakage detection | Detection of personal information leakage in Android applications | 2015 |
| | Song and Hengartner [140] | | Detection of information leakage in Android applications | 2015 |
| | Le et al. [90] | | Traffic collection and analysis to enable users have control over their data | 2015 |
| | Ren et al. [122] | | Cross-platform PII leaks identification giving users control over them | 2016 |
| | Continella et al. [51] | | PII leakage detection, resilient to obfuscation techniques | 2017 |
| | Rosner et al. [125] | | Information leaks in TLS-encrypted network traffic | 2019 |

traffic that is not encrypted, (iv) excess communication with more than expected sources, and finally (v) the communication of the majority of the examined apps with Google. TaintDroid [64] identifies privacy leaks of Android applications with dynamic information-flow tracking. More specifically, the authors monitored the behavior of popular third-party Android applications and discovered potential misuse cases of user private information across applications. TaintDroid provides users with information regarding third-party applications. Haystack [121] is a mobile application distributed via popular app stores that can correlate contextual information (such as application identifiers and radio state) with specific traffic flows (encrypted or not) destined to remote services. To handle encrypted traffic, haystack employs a transparent **man-in-the-middle** (**MITM**) proxy for TLS traffic. PrivacyGuard [140] is an open source platform that intercepts the network traffic that is generated by mobile applications (through VPN) in order to detect sensitive information leakage. PrivacyGuard is able to effectively detect information leakage in the majority of the applications under examination and it is shown that it outperforms TaintDroid. AntMonitor [90] is a system that passively monitors and collects packet-level measurements from Android devices in order to provide a fine-grained analysis. AntMonitor provides users with control over their personal data and supports client-side traffic collection and analysis. The authors examine PII-related features, such as (i) the IMEI, which uniquely identifies a device within a mobile network, and Android Device ID, which is an identification code associated with a device, and (ii) the phone number, email address, and location, which can be uniquely associated with users. ReCon [122] is a cross-platform system that reveals personally identifiable information leakages and gives users the control over leaked information without requiring any special privileges or custom OSs. ReCon uses machine learning to identify and reveal possible PII leakage by inspecting the network

traffic, while it provides a visualization tool that empowers users to control how their information is being handled by blocking or substituting it. Continella et al. [51] propose a system that detects leakages of PII and is resilient to obfuscation methods and techniques (e.g., encoding, formatting), encryption, or any other kind of transformation performed on private information before it is leaked. Moreover, Rosner et al. [125] present a black-box approach for detecting and quantifying side-channel information leaks in TLS-encrypted network traffic.

## 5.2 Fingerprinting

In this section, we present works in the field of webpage and application fingerprinting, even while covered by privacy enhancing technologies. In addition, we study the state-of-the-art in the device/OS identification and location estimation from network traffic inspection. The works that are presented in this section are also summarized in Table 8.

*5.2.1 Webpage Identification.* Traffic analysis has been used to identify fine-grained information, like webpages and websites, even while transferred over encrypted tunnels established by technologies such as OpenVPN [20] or TOR [24].

Herrmann et al. [75] present a classifier that identifies up to 97% of web requests from packet traces. Lu et al. [96] identify dynamic websites that are transferred over SSH and SSL tunnels. Panchenko et al. [111, 112] show how website fingerprinting in onion routing based anonymization networks, such as TOR, is still possible using information like packet sizes, total transmitted bytes, percentage of incoming packets, and others. Cai et al. [40] present a webpage fingerprinting attack that is resilient to recent traffic analysis countermeasures' methods, such as application-level defenses like HTTPOS [97] and randomized pipelining over TOR [118]. Kwon et al. [84] perform a passive attack against hidden services and their users called circuit fingerprinting attack. Using the attack, an attacker can identify the presence of hidden service activity in the network with high accuracy.

Draper-Gil et al. [60] study flow-based and time-related features that can be analyzed to detect VPN traffic and to classify encrypted traffic according to the type of traffic (e.g., browsing or streaming) using two machine learning techniques to test the accuracy of the features. They show that time-related features, when properly handled can reveal enough information for encrypted traffic characterization. Cruz et al. [55] present a deep learning method that takes advantage of a feature set that is based on the statistical behavior of TCP tunnels proxying BitTorrent traffic. The next steps are the transformation of this feature set into multiple timestep sequences and the training of a recurrent neural network. The results of this work show that it is possible to identify the existence of BitTorrent traffic in SSH tunnels.

Lotfollahi et al. [95] present a system that is able to handle both traffic characterization and application identification by analyzing encrypted traffic with deep learning. The proposed scheme can characterize between VPN and non-VPN traffic, protocols (e.g., FTP or P2P), and end-user applications (e.g., Skype or BitTorrent).

Shahbar and Zincir-Heywood [132] identify multilayer-encryption anonymity networks and the obfuscation techniques they use with a small number of features and a small number of packets. Montieri et al. [102] perform hierarchical traffic classification of anonymity tools, like TOR.

*5.2.2 Device/OS Identification.* There are many works that focus on extracting TCP or IP packet metadata, in order to investigate if the behavior of specific packet contents can be correlated with OSs, device types, and other characteristics.

Chen et al. [46] are able to perform OS identification, NAT, and tethering detection by examining multiple features in the TCP/IP packet headers. The authors use real network traffic traces to evaluate the accuracy of fingerprinting and show that several techniques that can successfully

fingerprint desktop OSs are not similarly effective for fingerprinting mobile devices. For OS fingerprinting, the authors use the following packet header values: the IP TTL value, the IP ID monotonicity, the TCP timestamp option, the TCP window size scale option, and the clock frequency. Features like the TCP timestamp monotonicity, clock frequency, and boot time can be used for tethering detection. Ruffing et al. [126] aim for OS identification of mobile devices even in encrypted traffic. The authors propose a traffic content agnostic algorithm that implements spectral analysis of the encrypted traffic and they show that even a network traffic input of 30 seconds can be enough for high accuracy results. Sivanathan et al. [136] classify IoT devices using network traffic features. More specifically, the authors instrument an ecosystem of different IoT devices (e.g., cameras, plugs, and motion sensors) and examine traffic characteristics, such as port numbers, signaling patterns, and cipher suites that are used. Lastovicka et al. [89] examine traffic patterns of the TLS protocol and train a machine learning model using features from the TLS handshake in order to identify the operating system of a device. They focus their research on mobile devices connected in a wireless network.

*5.2.3 Location Estimation.* The position of a mobile device can be calculated and estimated by collecting and monitoring the network traffic that is produced by applications that contain location-based services, even when the communication channels are encrypted. For example, Ateniese et al. [35] show that an adversary could estimate the position of a mobile device by analyzing the timing and sizes of encrypted network packets that are exchanged between the user's mobile device and any location-based service provider that communicates with the device.

### 5.3 Techniques

In this section, we present the techniques and algorithms that are more popular among the works of this category. Table 9 briefly describes the techniques and evaluation metrics used.

The majority of works in the category of website or device/OS fingerprinting domain use machine learning techniques, while it is also common to build fingerprints for a webpage and then compare its similarity for classification. Works that detect PII leakages either work offline (e.g., [90, 125]) or online (e.g., [51, 121]). Offline works use similar machine learning techniques as previously discussed works, while online tools intercept the encrypted network traffic before processing it.

For instance, between works that examine tunnelled network traffic (e.g., over VPN or SSH protocols) for website classification and fingerprinting, the most popular algorithms are Multinomial Naive Bayes [75], Support Vector Machine [112], and Hidden Markov Models [40]. In addition, Levenshtein distance and the Jaccard classifier are used in a number of works to examine similarities between website fingerprints and properly classify them into categories [40, 92, 96].

In addition, more recent works like [55, 95] use neural networks. Lotfollahi et al. [95] present a system that is able to handle both traffic characterization and application identification by analyzing encrypted traffic with deep learning, embedding stacked autoencoder, and CNN to classify network traffic. Cruz et al. [55] identify tunnelled BitTorrent traffic with a deep learning implementation. Their approach examines features that are related to the statistical behavior of TCP tunnels proxying BitTorrent traffic. Then, the authors transform the features into multiple time sequences and train a recurrent neural network. Profit combines techniques like network trace alignment, phase detection, feature selection, feature probability distribution estimation, and entropy computation to quantify the amount of information leakage that is revealed via the network traffic. Rosner et al. [125] present in "Profit" a dynamic technique to detect information leakages in applications that support encryption and communicate via TLS. Profit receives a "user-supplied profiling-input suite" where application data is annotated as secret or sensitive. Profit runs the

Table 9. Techniques and Algorithms Used by Works in the User Privacy Domain (Sorted by Category and Publication Year)

| Work | Category | Algorithm/Technique | Performance Evaluation Metrics |
|---|---|---|---|
| Herrmann et al. [75] | Webpage Identification | Classification with **Multinomial Naive Bayes (MNB)** | Accuracy, Training/Testing Time |
| Lu et al. [96] | | Fingerprint similarity with Levenstein distance | Accuracy, Effect of time on accuracy |
| Panchenko et al. [111, 112] | | Classification with SVM, Fingerprint similarity | Accuracy, FPR, TPR, Runtime, Recall, Precision |
| Cai et al. [40] | | Classification with SVM and Fingerprint similarity with Damerau-Levenshtein distance | Success rate, Likelihood, TPR |
| Kwon et al. [84] | | Classification with C4.5 Decision Trees and Fingerprint similarity with Edit Distance | Accuracy, TPR, FPR |
| Draper-Gil et al. [60] | | Classification with C4.5 and KNN | Precision, Recall, Accuracy |
| Cruz et al. [55] | | LSTM and BLSTM Deep Neural Networks | Precision, Recall, Accuracy, F1-score |
| Lotfollahi et al. [95] | | Deep Neural Network **Stacked Autoencoder (SAE)** and CNN | F1-score, TPR, FPR |
| Shahbar and Zincir-Heywood [132] | | Classification with C4.5 Decision Trees, RFs, Naive Bayes, **Bayesian Network (BN)** | Accuracy, Time |
| Montieri et al. [102] | | Classification with C4.5, RF, NB, BN | Accuracy, F-measure, G-mean |
| Chen et al. [46] | Device/OS Identification | Classification with Naive Bayes | Accuracy, Precision, Recall, F-measure |
| Sivanathan et al. [136] | | Classification with Naive Bayes Multinomial classifier, RF | Accuracy, **Root Relative Squared Error (RRSE)** |
| Skowron et al. [137] | | Classification with $k$-NN, Decision Trees, and RFs | Accuracy, F1-score, Precision, Recall |
| Lastovicka et al. [89] | | Classification with Decision Trees | Accuracy, Precision, Recall, F-measure |
| Ateniese et al. [35] | Location Estimation | Interception of encrypted traffic and payload inspection | Accuracy and Granularity of the monitor area |
| Razaghpanah et al. [121] | PII leakage Detection | Interception of encrypted traffic and payload inspection | CPU and Power Overhead, Latency, Throughput, TLS overhead |
| Song and Hengartner [140] | | Interception of encrypted traffic and payload inspection | Throughput, Delay, Battery Consumption |
| Le et al. [90] | | Interception of encrypted traffic and payload inspection, Classification with SVM | Precision, F1-score, CPU, and Battery cost |
| Ren et al. [122] | | Interception of encrypted traffic and payload inspection | Accuracy, Classification time, and Runtime |
| Continella et al. [51] | | Interception of encrypted traffic and payload inspection | False positives, Execution time |
| Rosner et al. [125] | | Trace Alignment, Phase Detection, Leakage Quantification with Shannon entropy | Information Leakage measure |

The table also includes the metrics that authors use to evaluate the work.

application over the user-supplied input and captures a set of variable-length network packet traces. The traces include information like packet sizes and timestamps along with their aggregations (e.g., total time and median size). Again, the authors agree that finding the features that leak the most information is challenging. To another end, device and OS identification techniques use network packet header contents, such as IP TTL value, the IP ID monotonicity, the TCP timestamp option, the TCP window size scale option, and the clock frequency. For the detection of tethering, similar approaches take into consideration the monotonicity of the TCP timestamp, the timestamp clock frequency, and the booting time [46].

Works that investigate PII leakage from mobile applications are also presented in Tables 8 and 9. In this category, works tend to perform network traffic interception to be able to process the encrypted traffic and follow the information flow that is exposed by mobile applications. After the traffic interception, the authors are able to extract information by inspecting plaintext packet contents.

### 5.4 Datasets

Table 10 contains the public datasets that were used by each work. Lastovicka et al. [89] produced a dataset that was made public after the publication of their work. The dataset contains TLS fingerprints collected. The VPN-nonVPN dataset (ISCXVPN2016) [11] contains traffic from user sessions in applications of browsing, email, chat, streaming, and others. The traffic is either regular or transferred over VPN. The LBNL/ICSI dataset [3] contains packet traces that span more than 100 hours of activity from a total of several thousand internal hosts. The Anon17 dataset [13] contains network traffic traces from TOR, JonDonym, and I2P anonymity tools. The CRAWDAD SIGCOMM

Table 10. Datasets Used by Works in the User Privacy Domain (Sorted by Category and Publication Year)

| Work | Category | Dataset Availability | Dataset details |
|---|---|---|---|
| Lotfollahi et al. [95] | Webpage Identification | Public | UNB ISCX VPN-nonVPN [11] |
| Shahbar and Zincir-Heywood [132] | | Public | LBNL/ICSI [3] |
| Montieri et al. [102] | | Public | Anon17 [13] |
| Chen et al. [46] | Device/OS Identification | Public | CRAWDAD SIGCOMM'08 [5], CRAWDAD OSDI'06 [4] |
| Lastovicka et al. [89] | | Proprietary dataset that was made public | TLS fingerprints for OS identification [17] |
| Rosner et al. [125] | PII Leakage Detection | Public | DARPA Space/Time Analysis for Cybersecurity program [9] |

'08 dataset [5] contains traces of wireless network activity from the SIGCOMM 2008 conference. Similarly, the CRAWDAD OSDI '06 dataset [4] contains network activity from the OSDI 2006 conference. Finally, the goal of the DARPA Space/Time Analysis for Cybersecurity program [9], among others, is to enable researchers to identify vulnerabilities related to the space and time resource usage behavior of algorithms.

### 5.5 Objectives and Limitations

In the categories of website, location, and device/OS identification, the major goal is to produce an effective methodology for accurate fingerprinting. Thus, all the works that we study in these categories have one primary target; functionality. On the other hand, works in the domain of PII leakage detection focus on the programmability and deployability. Hence, we encounter performance-driven solutions.

The limitation of high false-positive rates is major in any domain of traffic analysis. Unfortunately, it appears to be very difficult to produce a universal solution that will be able to cover a whole domain, due to the vast diversity and heterogeneity that has been introduced during recent years in every single aspect of th Internet. In addition, as Juarez et al. [78] argue regarding fingerprinting, accuracy scores reduce over time.

In addition, in the category of PII leakage detection, it is common to use proxy servers or VPN in order to redirect the traffic to a controlled environment for interception and decryption of encrypted traffic. Even if we neglect the latency that is added, this technique could eventually raise privacy-related concerns if users are not properly informed about the procedure of the processing and manipulation (e.g., storage) of their traffic and personal data. In Section 6, we discuss middleboxes that address this issue by processing sensitive information using hardware-assisted technologies for trusted execution.

## 6 NETWORK FUNCTIONS IN MIDDLEBOXES AFTER NETWORK ENCRYPTION

Quoting from RFC 3234, "a middlebox is defined as any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host" [42]. The typical use of a middlebox is to offer security (e.g., firewall, intrusion detection) or performance (e.g., caching, protocol accelerator), while other common uses are network address translation and protocol conversion. One challenge that occurred after the rapid growth of network encryption is that in order to process and operate on TLS traffic, the middlebox must perform a man-in-the-middle in a connection. Of course, this raises major concerns on user privacy preservation. Network middleboxes that aim to inspect encrypted traffic operate by acting as proxies. They terminate and decrypt the client-initiated TLS session, they analyze the HTTP plaintext content, and then they initiate a brand new TLS connection to the destination. TLS makes interception difficult by encrypting data and defending against attacks (like man-in-the-middle) via the certificate validation. During the validation process, the client is responsible to authenticate the identity of the destination server, rejecting impostors. To circumvent this

Table 11. Works that Implement Network Functions in Midlleboxes, Sorted by Publication Year

| Traffic Analysis Domain | Work | Goal | Year of Publication |
|---|---|---|---|
| Network Functions in Middleboxes | Sherry et al. [134] | DPI on the encrypted traffic using encrypted rules | 2015 |
| | Naylor et al. [106] | Extend the TLS protocol to support middleboxes | 2015 |
| | Asghar et al. [34] | Trusted execution for **Network Functions** (**NFs**) in the cloud | 2016 |
| | Canard et al. [41] | DPI on the encrypted traffic using encrypted rules | 2017 |
| | Lan et al. [87] | Trusted execution for NFs in the cloud | 2016 |
| | Yuan et al. [171] | DPI on the encrypted traffic using encrypted rules | 2016 |
| | Fan et al. [66] | DPI on the encrypted traffic using encrypted rules | 2017 |
| | Naylor et al. [105] | Secure outsourcing of middlebox NFs in untrusted infrastructures | 2017 |
| | Han et al. [74] | Secure outsourcing of middlebox NFs in untrusted infrastructures | 2017 |
| | Coughlin et al. [53] | Secure outsourcing of middlebox NFV in untrusted infrastructures | 2017 |
| | Poddar et al. [119] | Secure outsourcing of middlebox NFV in untrusted infrastructures | 2018 |
| | Trach et al. [148] | Secure outsourcing of middlebox NFs in untrusted infrastructures | 2018 |
| | Goltzsche et al. [71] | Secure **virtual private network** (**VPN**) with middlebox NF | 2018 |
| | Duan et al. [61] | Secure outsourcing of middlebox NFs in untrusted infrastructures | 2019 |
| | Ning et al. [107] | DPI on the encrypted traffic using encrypted rules | 2019 |
| | Guo et al. [72] | Privacy preserving packet header processing for middleboxes in the cloud | 2020 |

validation process, a self-signed CA certificate is injected into the client browser's root store at the time of installation. For network middleboxes, administrators deploy the middlebox certificate to the corresponding devices (e.g., of the organization) in a similar manner. Then, when the proxy intercepts a connection, it will dynamically generate a certificate for a website's domain name that is signed with its CA certificate. The proxy will deliver this certificate chain to the browser [62]. Some works that allow TLS interception to inspect encrypted network traffic have already been presented in Section 5.

In this section, we discuss the works that enable secure processing of encrypted traffic by middleboxes. Tables 11 and 12 present details about these works.

### 6.1 Network Functions in Middleboxes

BlindBox [134] performs deep-packet inspection directly on the encrypted traffic, utilizing a new protocol and new encryption schemes. Specifically, the functionality of BlindBox is provided through (i) a searchable encryption scheme [139] that enables the inspection of encrypted traffic for certain keywords, (ii) a technique to allow the middlebox to obtain encrypted rules (i.e., based on the rules from the middlebox and the private key of the endpoints), and (iii) a mechanism to allow flow decryption when a suspicious keyword is observed in the flow. mcTLS extends the traditional TLS protocol to support middleboxes by allowing endpoints and content providers to explicitly introduce middleboxes in secure end-to-end sessions while controlling which parts of the data they can read or write [106].

Asghar et al. [34] propose SplitBox, in which a cloud service provider privately computes network functions on behalf of the client. More specifically, SplitBox provides security guarantees in the honest-but-curious model and works based on cryptographic secret sharing. As proof-of-concept, the authors implemented a firewall and measured the bandwidth and latency achieved.

Embark [87] enables a cloud provider to support middlebox outsourcing respecting user privacy. Embark encrypts the traffic that is transmitted to the cloud and enables the cloud to process the encrypted traffic without having to decrypt it. Yuan et al. [171] propose a system architecture for outsourced middleboxes to perform DPI over encrypted traffic, without revealing either packet payloads or inspection rules. SPABox [66] is a middlebox-based system that supports keyword-based and data analysis–based DPI functions over encrypted traffic without having to decrypt it.

Canard et al. [41] present BlindIDS, which is able to perform deep packet inspection directly on encrypted network packets for intrusion detection. BlindIDS does not assume knowledge over the traffic content or the patterns of detection signatures. The authors evaluate the performance of BlindIDS by presenting the overhead on sender and receiver sides (i.e., connection setup time, data encryption time) and the overhead on the service provider (i.e., detection time, memory usage). Ning et al. [107] propose PrivDPI, a tool that addresses the performance limitations of BlindBox [134]. Guo et al. [72] propose a privacy preserving packet header processing approach for middleboxes that are outsourced to cloud infrastructures. The authors perform a security analysis and identify information leakages, while they evaluate the performance of the prototype (i.e., initialization time, memory cost, latency, throughput, and overhead).

To overcome the limitation of privacy violation when the network traffic is intercepted for further processing, there are works that propose hardware-assisted solutions that enable trusted execution. These works enable the secure processing of sensitive information, such as the network traffic, inside encrypted memory regions provided by **Trusted Execution Environments** (**TEEs**). One example of TEE is the Intel SGX technology, which is supported by Intel Skylake processors (and successors). TEEs offer secure processing when the execution environment could not be trusted (e.g., a cloud infrastructure).

Naylor et al. [105] propose mbTLS, a protocol that enables secure outsourcing middlebox functionality to untrusted infrastructure using the Intel SGX technology [8].

Han et al. [74] present SGX-Box, a secure middlebox system implementation that offers visibility in encrypted network traffic, taking advantage of the Intel's SGX technology [8]. SGX-Box ensures that the sensitive information, such as decrypted payloads and session keys, is securely protected within the protected memory enclave.

Coughlin et al. [53] propose the Intel SGX technology to overcome security issues of **Network Function Virtualization** (**NFV**) applications in cloud environments. Poddar et al. [119] present SafeBricks that also proposes the utilization of Intel SGX to shield the execution of NFV functions in untrusted environments like the cloud infrastructure. Similarly, Trach et al. [148] present ShieldBox, a middlebox framework for deploying network functions over untrusted commodity servers. ShieldBox takes advantage of the Intel SGX technology and the authors deploy two use cases: (i) a multiport IP router and (ii) an intrusion detection system. Goltzsche et al. [71] propose EndBox. EndBox executes middlebox functions on client machines at a network edge and combines a VPN with middlebox functions that are protected in Intel SGX hardware enclaves. Duan et al. [61] present LightBox, which offers efficient and protected middlebox functionality using the Intel SGX technology.

### 6.2 Techniques

Works in this category either process the encrypted network traffic using software-based cryptographic techniques, such as searchable encryption, or process the network traffic inside encrypted memory regions in order to ensure the preservation of privacy. With searchable encryption, tools are able to search directly on encrypted data without decrypting it, and thus, without leaking information in plaintext (e.g., for privacy preserving malware detection [56]). This is achieved by encrypting the rules to be searched against the already encrypted traffic (e.g., [41]). TEEs enable the secure code execution and information processing commonly using hardware-assisted features like memory enclaves. Intel SGX [8] is one of the most popular hardware-assisted TEEs and is frequently used to provide confidentiality and integrity guarantees to applications in environments where the OS could eventually become untrusted, like a cloud infrastructure (e.g., trusted antivirus in the cloud [57]).

Table 12. Techniques and Algorithms Used by Works that Implement Network Functions in Midlleboxes (Sorted by Publication Year)

| Work | Algorithm/Technique | Performance Evaluation Metrics |
|---|---|---|
| Sherry et al. [134] | DPI with Searchable Encryption | Overhead in Load Time and Bandwidth |
| Naylor et al. [106] | DPI with Searchable Encryption | Handshake, File Transfer, Page Load Times, Data Volume, CPU and Deployment Overheads |
| Asghar et al. [34] | DPI with Searchable Encryption | Throughput and Delay |
| Canard et al. [41] | DPI with Searchable Encryption | Connection Setup, Data Encryption and Detection Times, Memory Usage |
| Lan et al. [87] | DPI with Searchable Encryption | Performance Overhead in Throughput, Page Load Time, Time per-request |
| Yuan et al. [171] | DPI with Searchable Encryption | TPR and Initialization Time, Inspection Throughput, Token Overhead, Latency |
| Fan et al. [66] | DPI with Searchable Encryptionn | End-to-end Delay, Throughput, CPU Utilization, Connection Setup Overhead and Malware Detection Accuracy |
| Naylor et al. [105] | Secure hardware-assisted DPI (TEE: Intel SGX) | CPU Overhead, Handshake Latency, SGX I/O Throughput Overhead |
| Han et al. [74] | Secure hardware-assisted DPI (TEE: Intel SGX) | Performance Overhead in Throughput |
| Coughlin et al. [53] | Secure hardware-assisted DPI (TEE: Intel SGX) | Processing Throughput |
| Poddar et al. [119] | Secure hardware-assisted NFV (TEE: Intel SGX) | I/O Throughput and Memory Usage Overhead |
| Trach et al. [148] | Secure hardware-assisted DPI (TEE: Intel SGX) | Throughput, Latency, Scalability |
| Goltzsche et al. [71] | Secure hardware-assisted DPI (TEE: Intel SGX) | Round-Trip Time, Throughput, Latency, CPU usage Overheads |
| Duan et al. [61] | Secure hardware-assisted DPI (TEE: Intel SGX) | Throughput, CPU usage, Packet Delay |
| Ning et al. [107] | DPI with Searchable Encryption | Latency, Bandwidth, Token Encryption Time, Round-Trip Total Time |
| Guo et al. [72] | DPI with Searchable Encryption | Initialization Time, Memory Cost, Processing Latency, Throughput, Token Overhead |

The table also includes the metrics that authors use to evaluate the work.

In [34, 41, 66, 72, 87, 106, 107, 134, 171], the authors perform network middlebox applications with core functionality the deep packet inspection using searchable encryption. In [53, 61, 71, 74, 105, 119, 148] the authors shield the network traffic processing using hardware enclaves that the Intel SGX technology provides. Table 12 displays the most common techniques used for network functions in middleboxes. Also, we present the metrics that each work used for evaluation.

### 6.3 Datasets

The datasets used to evaluate the works of this section are not other than the ones used by traditional works in network traffic processing inside middleboxes. Depending on the application goal (e.g., intrusion detection or firewall), the authors utilize the relevant rules (e.g., Snort rules [22] in [41, 66, 107, 171]) and traffic traces (e.g., DARPA [1] in [171]).

In this section, we do not include a table, since the datasets used by works in this category do not provide labeled and encrypted network traffic to be used as ground-truth data for encrypted network traffic analysis.

### 6.4 Objectives and Limitations

While the main objective of the majority of the works presented in this survey is to ensure the functionality of their systems by providing knowledge on how to properly analyze encrypted network traffic in order to extract information about its nature, the works in this category focus also on the programmability and deployment of their systems. Thus, the authors provide evaluation results not only for the effectiveness of their solution but also for the processing performance and the overhead that is introduced using either a software-centric solution like the searchable encryption or a hardware-assisted technology like TEEs.

Although cryptographic tools like the searchable encryption are very effective and significantly preserve user privacy when it comes to network functions in middleboxes, the individual functions that are performed (e.g., rule encryption and connection setup) add an essential overhead to the end-to-end performance. Similarly, TEEs also increase the processing overhead. For instance, with Intel SGX, substantial overhead can be presented when the computations are I/O bound [43]. Even though both techniques are effective, they introduce the tradeoff of user privacy versus performance.

## 7 ENCRYPTED TRAFFIC ANALYSIS COUNTERMEASURES

As we discuss in this survey, numerous techniques have been employed in order to enable traffic processing when the network is encrypted or tunnelled. Even though encrypted traffic analysis techniques can be used by different actors (such as ISPs/CSPs, etc.) to *benignly* extract information about network usage, encrypted traffic analysis techniques can be also used by malicious actors (such as governments that censor websites or prohibit Internet usage) in order to harm the privacy that network encryption offers to Internet users. Some of the most popular solutions propose randomizing network packet sizes, padding bytes to packets for a fixed size and time tuning for inter-packet transmission. More sophisticated solutions are explicitly discussed in the following paragraphs. In this section, we discuss the (i) techniques that can be used against encrypted network traffic analysis and (ii) systems that exist and aim to defend against it.

### 7.1 Anonymity Tools

Onion routing serves as an overlay network designed to anonymize communications and applications (e.g., web browsing and instant messaging). TOR is one good example of onion routing. Among others, it uses fixed-size cells that are the unit of communication in TOR [24]. As we have already discussed in Section 5.2 though, anonymity tools are not enough to prevent website and hidden services identification [123], since the existence of anonymity tools can be identified using numerous encrypted traffic analysis techniques (Section 5.3).

### 7.2 Traffic Shaping

As already discussed, features and characteristics of network traffic that present patterns after encryption (e.g., packet sizes and timing), can reveal surprising information about the traffic's nature and contents. Even though encrypted traffic analysis can be legitimate, these techniques raise important concerns about privacy-related issues. An approach that typically mitigates such threats is the padding of packet sizes or the transmission of packets at fixed timing intervals; obfuscating the behavior of a communication mean. However, this method can become inefficient because it results in time overheads. Wright et al. propose a method for hindering statistical traffic analysis algorithms. Their approach proposes the modification of a certain network traffic "class" to look like another. The authors show how to modify packets' characteristics in real-time with low overhead in order to reduce the accuracy measurements of traffic classifiers. The morphed data is then sent to the network stack encrypted and then sent to the destination [166]. AnonRep [172] builds on top of anonymity and privacy guarantees for the case of reputation and voting systems. TARN [170] randomizes IP addresses, while TARANET [45] employs packet mixing and splitting to achieve constant-rate transmission, providing anonymity at the network layer. Luo et al. [97] design the HTTPOS fingerprinting defense at the application layer. HTTPOS acts as a proxy that receives HTTP requests and obfuscates them before allowing transmission. Specifically, it modifies network-related features, such as the total packet size, the packet timestamp, and the payload size. In addition, it uses HTTP pipelining to obfuscate the number of

the transmitted packets. The authors show that HTTPOS was successful in defending against a number of traffic classifiers. Dyer et al. [63] create a defense, namely, BuFLO that combines previously proposed countermeasures, such as fixed packet sizes and constant rate traffic. The authors improve other related defenses at the expense of a high bandwidth overhead. Cai et al. [39] make modifications to the BuFLO defense proposing the rate adaptation technique. Yet, this adds a bandwidth overhead. Nithyanand et al. [108] propose Glove, which groups website traffic into clusters. This provides privacy guarantees and reduces the bandwidth overhead by grouping web traffic into similar sets. Panchenko et al. [112] propose "website camouflage," which is actually an obfuscation technique that randomly requests a second website, simultaneously with the actually requested one. Frolov and Wustrow [67] propose uTLS that enables tool maintainers to automatically mimic other popular TLS implementations to prevent censorship. Walkie-Talkie is a website fingerprinting defense approach that modifies the browser to communicate in half-duplex mode, since it produces burst sequences that leak less information to the adversary. This makes sensitive and non-sensitive pages look the same [160].

*7.2.1 Traffic Analysis Resistant IoT Devices.* Hafeez et al. [73] demonstrate that an adversary, with access to the network traffic of a "smart" home network, can lead to the identification of the device types and some user interactions with IoT devices. In order to defend against traffic analysis attacks, the authors propose a "traffic morphing" technique that shapes network traffic in ways that make it more difficult to achieve an attack that identifies devices and activities. In order to mask the background traffic, the authors send traffic on an upstream link at a constant rate, irrespective of real background traffic rate of an IoT device. Meanwhile, when an IoT device is inactive, the authors send dummy traffic representing device activity to upstream link, so that an adversary cannot identify real activity of the IoT device. While this approach is not very sophisticated, it points to the direction of defending against traffic analysis on IoT devices.

*7.2.2 Traffic Analysis Resistant Messaging Applications.* There have been efforts to create messaging protocols that provide anonymity and privacy guarantees in the face of traffic analysis. Dissent [164] and Riposte [52] are systems that provide strong user privacy guarantees. They protect packet metadata, but they suffer from scalability issues. Herd [91] is another system that tackles the case of anonymity for VoIP calls, by addressing, like the former proposals, some of the limitations of the more general-purpose Tor anonymity network [59]. Vuvuzela [150] and Atom [85] are more scalable systems (thousands of messages for millions of users) that employ differential privacy to inject noise into observable metadata.

## 8 LESSONS LEARNED AND CONCLUSIONS

In this section, we summarize and review the findings that are present in the previous sections (Sections 3–6). As we have already mentioned, network traffic analysis and classification is very popular in many domains, such as network analytics, security, privacy, and fingerprinting. The proposed solutions could be divided using the techniques and methods used for encrypted traffic analysis and inspection into three categories: the works that perform (i) manipulation of traffic metadata and characteristics, (ii) interception of encrypted traffic, and (iii) utilization of cryptographic functions (see Figure 1).

Systems in the first category (i.e., manipulation of traffic metadata and characteristics) focus on investigating which network-related characteristics reveal the nature of the network traffic. Throughout all these works, we find similar methodologies that lay above different algorithms. As it occurs, we can divide the resulting network-related characteristics that make traffic

Table 13. Common Features Used to Reveal the Encrypted Network Traffic's Nature

| Feature Category | Feature |
|---|---|
| Packet-level | Packet size |
| | Packet payload size |
| | Packet direction |
| Connection-level | Flow bytes |
| | Flow packets |
| | TLS-handshake metadata |
| Time-related | Rate of incoming packets |
| | Rate of packets with payload |
| | Packet inter-arrival times |
| | Flow inter-arrival times |
| | Flow bytes per second |
| | Flow packets per second |
| | Flow duration |
| | Active flow duration |
| | Idle flow duration |

The table also includes the metrics that authors use to evaluate the work.

classification feasible—-even in encrypted networks—into three groups: (i) packet-level, (ii) connection-level, and (iii) time-related features. Of course, none of these characteristics include information extracted from packet payload contents, since it is assumed that network packets are encrypted. In Table 13, we present some of these characteristics (in the literature are also referred to as features, side-channel data, or/and metadata), which are either found inside packet headers or calculated. These features are the most widely used in the state-of-the-art for network analytics, security, privacy, and fingerprinting.

Systems in the second and third categories (i.e., interception of encrypted traffic and utilization of cryptographic functions), focus on proposing high-end techniques to offer efficient, adaptive, and scalable solutions for live and real-time network traffic inspection. The majority of such works are able to inspect the network and report findings, always aiming for good computational performance that can cope with current network speeds.

### 8.1 Limitations and Possible Future Directions

In this subsection, we review the state-of-the-art in network traffic analysis and inspection techniques by demonstrating their limitations.

In the first category *"manipulation of traffic metadata and characteristics,"* we encounter a very large number of works that examine the feasibility of identifying the network traffic class, while encrypted (e.g., a webpage, a mobile application, or some malicious activity). As already stressed, all these works focus on machine learning techniques that signify the network-related characteristics that indicate this specific class. Then, to validate and evaluate their approaches, they calculate metrics like accuracy, recall, and precision, comparing themselves to related works. However, all these works utilize a certain ground-truth dataset that enables them to train their models and eventually, teach them how to spot specific patterns inside the network traffic in order to locate the occurrence of a specific event (e.g., visiting a certain website). Even while scoring high performance (in metrics like accuracy, precision, and recall) in close-world scenarios (i.e., under a specific ground-truth dataset), these systems will certainly produce high rates of false positives when tested against real-world datasets of traffic traces; surely, encrypted network traffic will not enable this kind of evaluation from an ISP/CSP level. The limitation of high false-positive rates is major in any domain of traffic analysis. Unfortunately, it appears to be very difficult to

produce a universal solution that will be able to cover a whole domain, due to the vast diversity and heterogeneity that has been introduced during recent years in every single aspect of the Internet. In addition, as Juarez et al. [78] argue regarding fingerprinting, accuracy scores reduce over time.

In the second category *"interception of encrypted traffic,"* we encounter solutions that take advantage of behavioral patterns or signatures to report a specific event (e.g., malicious activity in a network). These solutions focus on the implementation of network inspection engines that perform directly on encrypted traffic that is actually intercepted and decrypted before the processing. This raises concerns about the privacy preservation and data integrity. In addition, decrypting and then re-encrypting network packets adds a significant end-to-end performance overhead. In order to offer privacy guarantees, numerous works propose hardware-assisted solutions, like TEEs. Using TEEs in untrusted infrastructures and OSs enables secure code execution in encrypted memory regions, something that offers strong privacy preserving guarantees. Of course, TEEs are prone to side-channel attacks [38, 161], while swapping into enclaves adds performance overhead, especially when applications are I/O intensive like network processing.

Finally, in the third category *"cryptographic functions,"* we explore works that use techniques like searchable encryption that enable the direct processing of encrypted data without having to decrypt it. These techniques are very effective in terms of privacy preservation, but they also add important performance overhead for real-time systems (such as network processing systems).

Apparently, there is a gap in the literature when it comes to real-time traffic processing when the network is encrypted. While there are numerous machine learning techniques that effectively classify the nature of the traffic, there is no option for online traffic processing. Furthermore, works that enable the online encrypted traffic inspection add considerable processing performance overheads that harm the concept of real-time processing in Gbps networks. Thus, the future research directions that will significantly enhance the state-of-the-art must meet the following requirements: (i) effectiveness in terms of classification accuracy in fully encrypted networks, (ii) reduction of false positives, (iii) good processing performance, and (iv) real-time processing in Gbps networks with reduced packet losses.

### 8.2 Conclusions

The increasing adoption of network encryption has introduced major challenges to typical deep packet inspection systems. Traditional deep packet inspection engines that have been widely used until recently for network analytics and security are not able to process network packets in the same way, since packet payloads are now getting encrypted. In this survey, we examine how network traffic processing systems are adapting to current encryption trends and how network traffic inspection systems process the traffic even though it is encrypted.

Indeed, after the thorough examination of the literature, it seems that the research community has already started finding solutions on how to bypass the constraints that are caused by network encryption to traditional deep packet inspection systems. Many works focus on the functionality of their solutions using machine or deep learning methods and algorithms, while others develop sophisticated techniques to process encrypted traffic directly in an online fashion. Yet, we discover that there is still considerable room for improvement. We present the limitations that we identify in the works that we survey, enabling researchers to recognize the possible future directions within the research area of encrypted traffic analysis and processing.

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and editors.

## REFERENCES


[1] 1999. The DARPA '99 Dataset. Retrieved February 1, 2021 from https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset.

[2] 1999. The KDD Cup 1999 Dataset. Retrieved February 1, 2021 from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[3] 2005. The LBNL/ICSI Dataset. Retrieved February 1, 2021 from https://www.icir.org/enterprise-tracing/download.html. Accessed: 2021-02-01.

[4] 2007. A Detailed Traceset of Network Activity at OSDI 2006. Retrieved February 1, 2021 from https://crawdad.org/microsoft/osdi2006/20070523/pcap/.

[5] 2008. Dataset of Wireless Network Measurement in the SIGCOMM 2008 Conference. Retrieved February 1, 2021 from https://crawdad.org/umd/sigcomm2008/20090302/.

[6] 2009. Application Layer Packet Classifier for Linux. Retrieved May 5, 2020 from http://l7-filter.sourceforge.net/.

[7] 2009. The NSL-KDD Dataset. Retrieved February 1, 2021 from https://www.unb.ca/cic/datasets/nsl.html.

[8] 2014. Intel Software Guard Extensions. Retrieved July 5, 2020 from https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html.

[9] 2015. Space/Time Analysis for Cybersecurity (STAC) . Retrieved February 1, 2021 from https://www.darpa.mil/program/space-time-analysis-for-cybersecurity.

[10] 2015. The UNSW-NB15 Dataset. Retrieved May 5, 2020 from https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/.

[11] 2016. The UNB ISCX VPN-nonVPN Dataset. Retrieved February 1, 2021 from https://www.unb.ca/cic/datasets/vpn.html.

[12] 2016. The USTC-TFC2016 Dataset. Retrieved February 1, 2021 from https://github.com/yungshenglu/USTC-TFC2016.

[13] 2017. The Anon17 Dataset. Retrieved February 1, 2021 from https://web.cs.dal.ca/~shahbar/data.html.

[14] 2018. The Recon Dataset. Retrieved February 1, 2021 from https://recon.meddle.mobi/appversions/.

[15] 2018. The Transport Layer Security (TLS) Protocol Version 1.3. Retrieved October 29, 2020 from https://tools.ietf.org/html/rfc8446.

[16] 2019. A Nonprofit Certificate Authority Providing TLS Certificates to 225 Million Websites.: 2019 Annual Report. Retrieved October 29, 2020 from https://abetterinternet.org/documents/2019-ISRG-Annual-Report-Desktop.pdf.

[17] 2019. Dataset Using TLS Fingerprints for OS Identification in Encrypted Traffic. Retrieved February 1, 2021 from https://zenodo.org/record/3461771.

[18] 2019. TLS 1.3: One Year Later. Retrieved October 29, 2020 from https://www.ietf.org/blog/tls13-adoption/.

[19] 2020. Let's Encrypt. Retrieved October 29, 2020 from https://letsencrypt.org.

[20] 2020. OpenVPN. Retrieved May 5, 2020 from https://openvpn.net.

[21] 2020. Suricata Open Source IDS/IPS/NSM Engine. Retrieved May 5, 2020 from https://www.suricata-ids.org/.

[22] 2020. The Snort IDS/IPS. Retrieved May 5, 2020 from https://www.snort.org/.

[23] 2020. The Zeek Network Security Monitor. Retrieved May 5, 2020 from https://www.zeek.org/.

[24] 2020. TOR Project. Retrieved from May 5, 2020 from https://www.torproject.org.

[25] 2021. The ISCX Dataset. Retrieved February 1, 2021 from https://www.unb.ca/cic/datasets/ids.html.

[26] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. 2018. Multi-classification approaches for classifying mobile app traffic. *Journal of Network and Computer Applications* 103 (2018), 131–145.

[27] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescape. 2019. MIMETIC: Mobile encrypted traffic classification using multimodal deep learning. *Computer Networks* 165 (2019), 106944.

[28] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescape. 2019. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Transactions on Network and Service Management* 16, 2 (2019), 445–458.

[29] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescape. 2020. Toward effective mobile encrypted traffic classification through deep learning. *Neurocomputing* 409 (2020), 306–315.

[30] Fabio Aiolli, Mauro Conti, Ankit Gangwal, and Mirko Polato. 2019. Mind your wallet's privacy: Identifying Bitcoin wallet apps and user's actions through network traffic analysis. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. ACM, 1484–1491.

[31] Hasan Faik Alan and Jasleen Kaur. 2016. Can Android applications be identified using only TCP/IP headers of their launch time traffic?. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 61–66.

[32] Payam Vahdani Amoli, Timo Hamalainen, Gil David, Mikhail Zolotukhin, and Mahsa Mirzamohammad. 2016. Unsupervised network intrusion detection systems for zero-day fast-spreading attacks and botnets. *JDCTA (International Journal of Digital Content Technology and its Applications* 10, 2 (2016), 1–13.

[33] Blake Anderson and David McGrew. 2017. Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1723–1732.

[34] Hassan Jameel Asghar, Luca Melis, Cyril Soldani, Emiliano De Cristofaro, Mohamed Ali Kaafar, and Laurent Mathy. 2016. Splitbox: Toward efficient private network function virtualization. In *Proceedings of the 2016 Workshop on Hot Tpics in Middleboxes and Network Function Virtualization*. 7–13.

[35] Giuseppe Ateniese, Briland Hitaj, Luigi Vincenzo Mancini, Nino Vincenzo Verde, and Antonio Villani. 2015. No place to hide that bytes won't reveal: Sniffing location-based encrypted traffic to track a user's position. In *International Conference on Network and System Security*. Springer, 46–59.

[36] Laurent Bernaille and Renata Teixeira. 2007. Early recognition of encrypted applications. In *International Conference on Passive and Active Network Measurement*. Springer, 165–175.

[37] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. 2006. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review* 36, 2 (2006), 23–26.

[38] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. 2017. Software grand exposure:SGX cache attacks are practical. In *11th USENIX Workshop on Offensive Technologies (WOOT'17)*.

[39] Xiang Cai, Rishab Nithyanand, and Rob Johnson. 2014. CS-BuFLO: A congestion sensitive website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 121–130.

[40] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM conference on Computer and Communications Security*. ACM, 605–616.

[41] Sebastien Canard, Aida Diop, Nizar Kheir, Marie Paindavoine, and Mohamed Sabt. 2017. BlindIDS: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. 561–574.

[42] Brian Carpenter and Scott Brim. 2002. *Middleboxes: Taxonomy and Issues*. Technical Report. RFC 3234.

[43] Sang Kil Cha, Iulian Moraru, Jiyong Jang, John Truelove, David Brumley, and David G. Andersen. 2011. SplitScreen: Enabling efficient, distributed malware detection. *Journal of Communications and Networks* 13, 2 (2011), 187–200.

[44] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)* 41, 3 (2009), 1–58.

[45] Chen Chen, Daniele E Asoni, Adrian Perrig, David Barrera, George Danezis, and Carmela Troncoso. 2018. TARANET: Traffic-analysis resistant anonymity at the NETwork layer. Retrieved May 21, 2021 from https://arxiv.org/abs/1802.09089.

[46] Yi-Chao Chen, Yong Liao, Mario Baldi, Sung-Ju Lee, and Lili Qiu. 2014. OS fingerprinting and tethering detection in mobile networks. In *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 173–180.

[47] C. R. Clark, Wenke Lee, D. E. Schimmel, Didier Contis, Mohamed Koné, and Ashley Thomas. 2005. A hardware platform for network intrusion detection and prevention. In *Proceedings of the 3rd Workshop on Network Processors and Applications (NP3'05)*.

[48] Mauro Conti, Qian Qian Li, Alberto Maragno, and Riccardo Spolaor. 2018. The dark side (-channel) of mobile devices: A survey on network traffic analysis. *IEEE Communications Surveys & Tutorials* 20, 4 (2018), 2658–2713.

[49] Mauro Conti, Luigi V. Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. 2015. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*. ACM, 297–304.

[50] Mauro Conti, Luigi Vincenzo Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. 2016. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions on Information Forensics and Security* 11, 1 (2016), 114–125.

[51] Andrea Continella, Yanick Fratantonio, Martina Lindorfer, Alessandro Puccetti, Ali Zand, Christopher Kruegel, and Giovanni Vigna. 2017. Obfuscation-resilient privacy leak detection for mobile apps through differential analysis. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS'17)*. 1–16.

[52] Henry Corrigan-Gibbs, Dan Boneh, and David Mazieres. 2015. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy (SP'15)*. IEEE, 321–338.

[53] Michael Coughlin, Eric Keller, and Eric Wustrow. 2017. Trusted click: Overcoming security issues of NFV in the cloud. In *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. 31–36.

[54] Scott E. Coull and Kevin P. Dyer. 2014. Traffic analysis of encrypted messaging services: Apple imessage and beyond. *ACM SIGCOMM Computer Communication Review* 44, 5 (2014), 5–11.

[55] Michelangelo Cruz, Roel Ocampo, Isabel Montes, and Rowel Atienza. 2017. Fingerprinting bittorrent traffic in encrypted tunnels using recurrent deep learning. In *2017 5th International Symposium on Computing and Networking (CANDAR'17)*. IEEE, 434–438.

[56] Helei Cui, Yajin Zhou, Cong Wang, Qi Li, and Kui Ren. 2018. Towards privacy-preserving malware detection systems for android. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS'18)*. IEEE, 545–552.
[57] Dimitris Deyannis, Eva Papadogiannaki, Giorgos Kalivianakis, Giorgos Vasiliadis, and Sotiris Ioannidis. 2020. Trustav: Practical and privacy preserving malware analysis in the cloud. In *Proceedings of the 10th ACM Conference on Data and Application Security and Privacy*. 39–48.
[58] Giorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, and Konstantina Papagiannaki. 2016. Measuring video QoE from encrypted traffic. In *Proceedings of the 2016 Internet Measurement Conference*. 513–526.
[59] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The second-generation onion router. In *13th USENIX Security Symposium (USENIX Security'04)*. 303–320.
[60] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. 2016. Characterization of encrypted and VPN traffic using time-related features. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP'16)*. 407–414.
[61] Huayi Duan, Cong Wang, Xingliang Yuan, Yajin Zhou, Qian Wang, and Kui Ren. 2019. Lightbox: Full-stack protected stateful middlebox at lightning speed. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2351–2367.
[62] Zakir Durumeric, Zane Ma, Drew Springall, Richard Barnes, Nick Sullivan, Elie Bursztein, Michael Bailey, J. Alex Halderman, and Vern Paxson. 2017. The Security Impact of HTTPS interception. In *NDSS*.
[63] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail. In *2012 IEEE Symposium on Security and Privacy*. IEEE, 332–346.
[64] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. 2014. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)* 32, 2 (2014), 5.
[65] Zubair Md Fadlullah, Tarik Taleb, Nirwan Ansari, Kazuo Hashimoto, Yutaka Miyake, Yoshiaki Nemoto, and Nei Kato. 2007. Combating against attacks on encrypted protocols. In *2007 IEEE International Conference on Communications*. IEEE, 1211–1216.
[66] Jingyuan Fan, Chaowen Guan, Kui Ren, Yong Cui, and Chunming Qiao. 2017. SPABox: Safeguarding privacy during deep packet inspection at a middlebox. *IEEE/ACM Transactions on Networking (TON)* 25, 6 (2017), 3753–3766.
[67] Sergey Frolov and Eric Wustrow. 2019. The use of TLS in censorship circumvention. In *NDSS*.
[68] Yanjie Fu, Hui Xiong, Xinjiang Lu, Jin Yang, and Can Chen. 2016. Service usage classification with encrypted internet traffic in mobile messaging apps. *IEEE Transactions on Mobile Computing* 15, 11 (2016), 2851–2864.
[69] Giannis Giakoumakis, Eva Papadogiannaki, Giorgos Vasiliadis, and Sotiris Ioannidis. 2020. Pythia: Scheduling of concurrent network packet processing applications on heterogeneous devices. In *2020 6th IEEE Conference on Network Softwarization (NetSoft'20)*. IEEE, 145–149.
[70] Younghwan Go, Muhammad Asim Jamshed, YoungGyoun Moon, Changho Hwang, and KyoungSoo Park. 2017. APUNet: Revitalizing GPU as packet processing accelerator. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17)*. 83–96.
[71] David Goltzsche, Signe Rusch, Manuel Nieke, Sebastien Vaucher, Nico Weichbrodt, Valerio Schiavoni, Pierre-Louis Aublin, Paolo Cosa, Christof Fetzer, Pascal Felber, et al. 2018. Endbox: Scalable middlebox functions using client-side trusted execution. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'18)*. IEEE, 386–397.
[72] Yu Guo, Mingyue Wang, Cong Wang, Xingliang Yuan, and Xiaohua Jia. 2020. Privacy-preserving packet header checking over in-the-cloud middleboxes. *IEEE Internet of Things Journal* 7, 6 (2020), 5359–5370.
[73] Ibbad Hafeez, Markku Antikainen, and Sasu Tarkoma. 2019. Protecting IoT-environments against traffic analysis attacks with traffic morphing. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops'19)*. IEEE, 196–201.
[74] Juhyeng Han, Seongmin Kim, Jaehyeong Ha, and Dongsu Han. 2017. SGX-Box: Enabling visibility on encrypted traffic using a secure middlebox module. In *Proceedings of the First Asia-Pacific Workshop on Networking*. ACM, 99–105.
[75] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. 2009. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial Naïve-Bayes classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*. 31–42.
[76] Nen-Fu Huang, Hsien-Wei Hung, Sheng-Hung Lai, Yen-Ming Chu, and Wen-Yen Tsai. 2008. A GPU-based multiple-pattern matching algorithm for network intrusion detection systems. In *22nd International Conference on Advanced Information Networking and Applications-Workshops (AINA Workshops 2008)*. IEEE, 62–67.
[77] Minghao Jiang, Gaopeng Gou, Junzheng Shi, and Gang Xiong. 2019. I know what you are doing with remote desktop. In *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC'19)*. IEEE, 1–7.

[78] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. 2014. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 263–274.

[79] Thomas Karagiannis, Andre Broido, Nevil Brownlee, Kimberly C. Claffy, and Michalis Faloutsos. 2004. Is P2P dying or just hiding? [P2P traffic measurement]. In *IEEE Global Telecommunications Conference (GLOBECOM'04)*, Vol. 3. IEEE, 1532–1538.

[80] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. 2005. BLINC: Multilevel traffic classification in the dark. In *ACM SIGCOMM Computer Communication Review*, Vol. 35. ACM, 229–240.

[81] Jawad Khalife, Amjad Hajjar, and Jesus Diaz-Verdejo. 2014. A multilevel taxonomy and requirements for an optimal traffic-classification model. *International Journal of Network Management* 24, 2 (2014), 101–120.

[82] Muhammad Jawad Khokhar, Thibaut Ehlinger, and Chadi Barakat. 2019. From network traffic measurements to QoE for internet video. In *2019 IFIP Networking Conference (IFIP Networking)*. IEEE, 1–9.

[83] Andreas Kind, Marc Ph. Stoecklin, and Xenofontas Dimitropoulos. 2009. Histogram-based traffic anomaly detection. *IEEE Transactions on Network and Service Management* 6, 2 (2009), 110–121.

[84] Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. 2015. Circuit fingerprinting attacks: Passive deanonymization of Tor hidden services. In *24th USENIX Security Symposium (USENIX Security'15)*. 287–302.

[85] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. 2017. Atom: Horizontally scaling strong anonymity. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 406–422.

[86] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C. Suh, Ikkyun Kim, and Kuinam J. Kim. 2019. A survey of deep learning-based network anomaly detection. *Cluster Computing* 22 (2019), 949–961.

[87] Chang Lan, Justine Sherry, Raluca Ada Popa, Sylvia Ratnasamy, and Zhi Liu. 2016. Embark: Securely outsourcing middleboxes to the cloud. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI'16)*. 255–273.

[88] Arash Habibi Lashkari, Andi Fitriah A. Kadir, Hugo Gonzalez, Kenneth Fon Mbah, and Ali A. Ghorbani. 2017. Towards a network-based framework for android malware detection and characterization. In *2017 15th Annual Conference on Privacy, Security and Trust (PST'17)*. IEEE, 233–234.

[89] Martin Lavstovivcka, Stanislav Spavcek, Petr Velan, and Pavel Celeda. 2020. Using TLS fingerprints for OS identification in encrypted traffic. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–6.

[90] Anh Le, Janus Varmarken, Simon Langhoff, Anastasia Shuba, Minas Gjoka, and Athina Markopoulou. 2015. AntMonitor: A system for monitoring from mobile devices. In *Proceedings of the 2015 ACM SIGCOMM Workshop on Crowdsourcing and Crowdsharing of Big (Internet) Data*. ACM, 15–20.

[91] Stevens Le Blond, David Choffnes, William Caldwell, Peter Druschel, and Nicholas Merritt. 2015. Herd: A scalable, traffic analysis resistant anonymity network for VoIP systems. In *ACM SIGCOMM Computer Communication Review*, Vol. 45. ACM, 639–652.

[92] Marc Liberatore and Brian Neil Levine. 2006. Inferring the source of encrypted HTTP connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*. ACM, 255–263.

[93] Junming Liu, Yanjie Fu, Jingci Ming, Yong Ren, Leilei Sun, and Hui Xiong. 2017. Effective and real-time in-app activity analysis in encrypted internet traffic streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 335–344.

[94] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. 2017. Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE Access* 5 (2017), 18042–18050.

[95] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. 2017. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing* 24 (2020), 1999–2012.

[96] Liming Lu, Ee-Chien Chang, and Mun Choon Chan. 2010. Website fingerprinting and identification using ordered feature sequences. In *European Symposium on Research in Computer Security*. Springer, 199–214.

[97] Xiapu Luo, Peng Zhou, Edmond W. W. Chan, Wenke Lee, Rocky K. C. Chang, and Roberto Perdisci. 2011. HTTPOS: Sealing information leaks with browser-side obfuscation of encrypted flows. In *NDSS*, Vol. 11. Citeseer.

[98] M. Hammad Mazhar and Zubair Shafiq. 2018. Real-time video quality of experience monitoring for HTTPS and QUIC. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 1331–1339.

[99] Chad R. Meiners, Jignesh Patel, Eric Norige, Eric Torng, and Alex X. Liu. 2010. Fast regular expression matching using small TCAMs for network intrusion detection and prevention systems. In *Proceedings of the 19th USENIX Conference on Security*. USENIX Association, 8–8.

[100] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: An ensemble of autoencoders for online network intrusion detection. Retrieved May 21, 2021 from https://arxiv.org/abs/1802.09089.

[101] AysŞe Rumeysa Mohammed, Shady A. Mohammed, and Shervin Shirmohammadi. 2019. Machine learning and deep learning based traffic classification and prediction in software defined networking. In *2019 IEEE International Symposium on Measurements & Networking (M&N'19)*. IEEE, 1–6.

[102] Antonio Montieri, Domenico Ciuonzo, Giampaolo Bovenzi, Valerio Persico, and Antonio Pescapé. 2019. A Dive into the Dark Web: Hierarchical traffic classification of anonymity tools. *IEEE Transactions on Network Science and Engineering* (2019).

[103] Andrew W. Moore and Denis Zuev. 2005. Internet traffic classification using Bayesian analysis techniques. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 33. ACM, 50–60.

[104] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS'15)*. IEEE, 1–6.

[105] David Naylor, Richard Li, Christos Gkantsidis, Thomas Karagiannis, and Peter Steenkiste. 2017. And then there were more: Secure communication for more than two parties. In *Proceedings of the 13th International Conference on Emerging Networking Experiments and Technologies*. ACM, 88–100.

[106] David Naylor, Kyle Schomp, Matteo Varvello, Ilias Leontiadis, Jeremy Blackburn, Diego R. Lopez, Konstantina Papagiannaki, Pablo Rodriguez Rodriguez, and Peter Steenkiste. 2015. Multi-context TLS (mcTLS): Enabling secure in-network functionality in TLS. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 199–212.

[107] Jianting Ning, Geong Sen Poh, Jia-Chng Loh, Jason Chia, and Ee-Chien Chang. 2019. PrivDPI: Privacy-preserving encrypted traffic inspection with reusable obfuscated rules. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1657–1670.

[108] Rishab Nithyanand, Xiang Cai, and Rob Johnson. 2014. Glove: A bespoke website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 131–134.

[109] Quamar Niyaz, Weiqing Sun, and Ahmad Y. Javaid. 2016. A deep learning based DDoS detection system in software-defined networking (SDN'16). Retrieved May 21, 2021 from https://arxiv.org/abs/1611.07400.

[110] Irena Orsolic, Dario Pevec, Mirko Suznjevic, and Lea Skorin-Kapov. 2016. YouTube QoE estimation based on the analysis of encrypted network traffic using machine learning. In *2016 IEEE Globecom Workshops (GC Wkshps'16)*. IEEE, 1–6.

[111] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website fingerprinting at internet scale. In *NDSS*.

[112] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*. ACM, 103–114.

[113] Eva Papadogiannaki, Dimitris Deyannis, and Sotiris Ioannidis. 2020. Head (er) Hunter: Fast intrusion detection using packet metadata signatures. In *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD'20)*. IEEE, 1–6.

[114] Eva Papadogiannaki, Constantinos Halevidis, Periklis Akritidis, and Lazaros Koromilas. 2018. OTTer: A scalable high-resolution encrypted traffic identification engine. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 315–334.

[115] Eva Papadogiannaki and Sotiris Ioannidis. 2021. Acceleration of intrusion detection in encrypted network traffic using heterogeneous hardware. *Sensors* 21, 4 (2021), 1140.

[116] Eva Papadogiannaki, Lazaros Koromilas, Giorgos Vasiliadis, and Sotiris Ioannidis. 2017. Efficient software packet processing on heterogeneous and asymmetric hardware architectures. *IEEE/ACM Transactions on Networking* 25, 3 (2017), 1593–1606.

[117] Vern Paxson, Robin Sommer, and Nicholas Weaver. 2007. An architecture for exploiting multi-core processors to parallelize network intrusion prevention. In *2007 IEEE Sarnoff Symposium*. IEEE, 1–7.

[118] Mike Perry. 2011. Experimental defense for website traffic fingerprinting. *Tor project blog*. Retrieved on May 21, 2021 from https://blog.torproject.org/experimental-defense-website-traffic-fingerprinting.

[119] Rishabh Poddar, Chang Lan, Raluca Ada Popa, and Sylvia Ratnasamy. 2018. Safebricks: Shielding network functions in the cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI'18)*. 201–216.

[120] Marianna Rapoport, Philippe Suter, Erik Wittern, Ondrej Lhotak, and Julian Dolby. 2017. Who you gonna call? Analyzing web requests in Android applications. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR'17)*. IEEE, 80–90.

[121] Abbas Razaghpanah, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Christian Kreibich, Phillipa Gill, Mark Allman, and Vern Paxson. 2015. Haystack: In situ mobile traffic analysis in user space. Retrieved May 21, 2021 from https://arxiv.org/abs/1510.01419.

[122] Jingjing Ren, Ashwin Rao, Martina Lindorfer, Arnaud Legout, and David Choffnes. 2016. ReCon: Revealing and controlling PII leaks in mobile network traffic. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 361–374.

[123] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. 2017. Automated website fingerprinting through deep learning. Retrieved May 21, 2021 from https://arxiv.org/abs/1708.06376.

[124] Luigi Rizzo, Marta Carbone, and Gaetano Catalli. 2012. Transparent acceleration of software packet forwarding using netmap. In *2012 Proceedings IEEE INFOCOM*. IEEE, 2471–2479.

[125] Nicolás Rosner, Ismet Burak Kadron, Lucas Bang, and Tevfik Bultan. 2019. Profit: Detecting and quantifying side channels in networked applications.. In *NDSS*.

[126] Nicholas Ruffing, Ye Zhu, Rudy Libertini, Yong Guan, and Riccardo Bettati. 2016. Smartphone reconnaissance: Operating system identification. In *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC'16)*. IEEE, 1086–1091.

[127] Ola Salman, Imad H. Elhajj, Ayman Kayssi, and Ali Chehab. 2020. A review on machine learning–based approaches for Internet traffic classification. *Annals of Telecommunications* 75, 11 (2020), 673–710.

[128] Brendan Saltaformaggio, Hongjun Choi, Kristen Johnson, Yonghwi Kwon, Qi Zhang, Xiangyu Zhang, Dongyan Xu, and John Qian. 2016. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic. In *WOOT*.

[129] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. 2017. Beauty and the burst: Remote identification of encrypted video streams. In *26th USENIX Security Symposium (USENIX Security 17)*. 1357–1374.

[130] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. 2012. Andromaly: A behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems* 38, 1 (2012), 161–190.

[131] Asaf Shabtai, Lena Tenenboim-Chekina, Dudu Mimran, Lior Rokach, Bracha Shapira, and Yuval Elovici. 2014. Mobile malware detection through analysis of deviations in application network behavior. *Computers & Security* 43 (2014), 1–18.

[132] Khalid Shahbar and A. Nur Zincir-Heywood. 2017. Packet momentum for identification of anonymity networks. *Journal of Cyber Security and Mobility* 6, 1 (2017), 27–56.

[133] Meng Shen, Jinpeng Zhang, Liehuang Zhu, Ke Xu, Xiaojiang Du, and Yiting Liu. 2019. Encrypted traffic classification of decentralized applications on ethereum using feature fusion. In *Proceedings of the International Symposium on Quality of Service*. ACM, 18.

[134] Justine Sherry, Chang Lan, Raluca Ada Popa, and Sylvia Ratnasamy. 2015. Blindbox: Deep packet inspection over encrypted traffic. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 213–226.

[135] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. 2018. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence* 2, 1 (2018), 41–50.

[136] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2018. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing* 18, 8 (2018), 1745–1759.

[137] Monika Skowron, Artur Janicki, and Wojciech Mazurczyk. 2020. Traffic fingerprinting attacks on internet of things using machine learning. *IEEE Access* 8 (2020), 20386–20400.

[138] Randy Smith, Neelam Goyal, Justin Ormont, Karthikeyan Sankaralingam, and Cristian Estan. 2009. Evaluating GPUs for network packet signature matching. In *2009 IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE, 175–184.

[139] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. 2000. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P 2000)*. IEEE, 44–55.

[140] Yihang Song and Urs Hengartner. 2015. Privacyguard: A VPN-based platform to detect information leakage on android devices. In *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*. ACM, 15–26.

[141] Ioannis Sourdis and Dionisios Pnevmatikatos. 2004. Pre-decoded CAMs for efficient and high-speed NIDS pattern matching. In *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*. IEEE, 258–267.

[142] Nasrin Sultana, Naveen Chilamkurti, Wei Peng, and Rabei Alhadad. 2019. Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications* 12, 2 (2019), 493–501.

[143] Hamid Tahaei, Firdaus Afifi, Adeleh Asemi, Faiz Zaki, and Nor Badrul Anuar. 2020. The rise of traffic classification in IoT networks: A survey. *Journal of Network and Computer Applications* 154 (2020), 102538.

[144] Tarik Taleb, Zubair Md. Fadlullah, Kazuo Hashimoto, Yoshiaki Nemoto, and Nei Kato. 2007. Tracing back attacks against encrypted protocols. In *Proceedings of the 2007 International Conference on Wireless Communications and Mobile Computing*. ACM, 121–126.

[145] Tuan A. Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. 2016. Deep learning approach for network intrusion detection in software defined networking. In *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM'16)*. IEEE, 258–263.

[146] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2016. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P'16)*. IEEE, 439–454.

[147] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2018. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security* 13, 1 (2018), 63–78.

[148] Bohdan Trach, Alfred Krohmer, Franz Gregor, Sergei Arnautov, Pramod Bhatotia, and Christof Fetzer. 2018. Shieldbox: Secure middleboxes using shielded execution. In *Proceedings of the Symposium on SDN Research*. 1–14.

[149] Matthias Vallentin, Robin Sommer, Jason Lee, Craig Leres, Vern Paxson, and Brian Tierney. 2007. The NIDS cluster: Scalable, stateful network intrusion detection on commodity hardware. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 107–126.

[150] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. 2015. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM, 137–152.

[151] Thijs van Ede, Riccardo Bortolameotti, Andrea Continella, Jingjing Ren, Daniel J. Dubois, Martina Lindorfer, David Choffnes, Maarten van Steen, and Andreas Peter. 2020. FLOWPRINT: Semi-supervised mobile-app fingerprinting on encrypted network traffic. In *NDSS*.

[152] Giorgos Vasiliadis, Spiros Antonatos, Michalis Polychronakis, Evangelos P. Markatos, and Sotiris Ioannidis. 2008. Gnort: High performance network intrusion detection using graphics processors. In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*.

[153] Giorgos Vasiliadis, Lazaros Koromilas, Michalis Polychronakis, and Sotiris Ioannidis. 2014. GASPP: A GPU-accelerated stateful packet processing framework. In *Proceedings of the 2014 USENIX Annual Technical Conference*.

[154] Giorgos Vasiliadis, Michalis Polychronakis, Spiros Antonatos, Evangelos P. Markatos, and Sotiris Ioannidis. 2009. Regular expression matching on graphics hardware for intrusion detection. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*.

[155] Giorgos Vasiliadis, Michalis Polychronakis, and Sotiris Ioannidis. 2011. MIDeA: A Multi-parallel intrusion detection architecture. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*.

[156] Petr Velan, Milan Čermák, Pavel Čeleda, and Martin Drašar. 2015. A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management* 25, 5 (2015), 355–374.

[157] Pan Wang, Xuejiao Chen, Feng Ye, and Zhixin Sun. 2019. A survey of techniques for mobile service encrypted traffic classification using deep learning. *IEEE Access* 7 (2019), 54024–54033.

[158] Qinglong Wang, Amir Yahyavi, Bettina Kemme, and Wenbo He. 2015. I know what you did on your smartphone: Inferring app usage over encrypted data traffic. In *2015 IEEE Conference on Communications and Network Security (CNS'15)*, IEEE, 433–441.

[159] Shanshan Wang, Zhenxiang Chen, Lei Zhang, Qiben Yan, Bo Yang, Lizhi Peng, and Zhongtian Jia. 2016. TrafficAV: An effective and explainable detection of mobile malware behavior using network traffic. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS'16)*. IEEE, 1–6.

[160] Tao Wang and Ian Goldberg. 2017. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *26th USENIX Security Symposium (USENIX Security'17)*. 1375–1390.

[161] Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, XiaoFeng Wang, Vincent Bindschaedler, Haixu Tang, and Carl A. Gunter. 2017. Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2421–2434.

[162] Yaru Wang, Ning Zheng, Ming Xu, Tong Qiao, Qiang Zhang, Feipeng Yan, and Jian Xu. 2019. Hierarchical identifier: Application to user privacy eavesdropping on mobile payment app. *Sensors* 19, 14 (2019), 3052.

[163] Xuetao Wei, Lorenzo Gomez, Iulian Neamtiu, and Michalis Faloutsos. 2012. ProfileDroid: Multi-layer profiling of android applications. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*. ACM, 137–148.

[164] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. 2012. Dissent in Numbers: Making strong anonymity scale. In *OSDI*. 179–182.

[165] Charles V. Wright, Lucas Ballard, Scott E. Coull, Fabian Monrose, and Gerald M. Masson. 2008. Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations. In *IEEE Symposium on Security and Privacy (SP'08)*. IEEE, 35–49.

[166] Charles V. Wright, Scott E. Coull, and Fabian Monrose. 2009. Traffic morphing: An efficient defense against statistical traffic analysis.. In *NDSS*, Vol. 9. Citeseer.

[167] Shichang Xu, Subhabrata Sen, and Z. Morley Mao. 2020. CSI: Inferring mobile ABR video adaptation behavior under HTTPS and QUIC. In *Proceedings of the 15th European Conference on Computer Systems*. 1–16.

[168] Haipeng Yao, Pengcheng Gao, Jingjing Wang, Peiying Zhang, Chunxiao Jiang, and Zhu Han. 2019. Capsule network assisted IoT traffic classification mechanism for smart cities. *IEEE Internet of Things Journal* 6, 5 (2019), 7515–7525.

[169] Fang Yu, Randy H. Katz, and Tirunellai V. Lakshman. 2004. Gigabit rate packet pattern-matching using TCAM. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04)*. IEEE, 174–183.

[170] Lu Yu, Qing Wang, Geddings Barrineau, Jon Oakley, Richard R. Brooks, and Kuang-Ching Wang. 2017. TARN: A SDN-based traffic analysis resistant network architecture. Retrieved May 21, 2021 from https://arxiv.org/abs/1709.00782.

[171] Xingliang Yuan, Xinyu Wang, Jianxiong Lin, and Cong Wang. 2016. Privacy-preserving deep packet inspection in outsourced middleboxes. In *The 35th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM'16)*. IEEE, 1–9.

[172] Ennan Zhai, David Isaac Wolinsky, Ruichuan Chen, Ewa Syta, Chao Teng, and Bryan Ford. 2016. AnonRep: Towards tracking-resistant anonymous reputation. In *NSDI*. 583–596.

[173] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. 2019. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys & Tutorials* 21, 3 (2019), 2224–2287.

Received August 2020; revised February 2021; accepted March 2021