



Optimal Packet Camouflage Against Traffic Analysis

LOUMA CHADDAD, ALI CHEHAB, IMAD H. ELHAJJ, and AYMAN KAYSSI,
Department Electrical and Computer Engineering, American University of Beirut

Research has proved that supposedly secure encrypted network traffic is actually threatened by privacy and security violations from many aspects. This is mainly due to flow features leaking evidence about user activity and data content. Currently, adversaries can use statistical traffic analysis to create classifiers for network applications and infer users' sensitive data. In this article, we propose a system that optimally prevents traffic feature leaks. In our first algorithm, we model the packet length probability distribution of the source app to be protected and that of the target app that the source app will resemble. We define a model that mutates the packet lengths of a source app to those lengths from the target app having similar bin probability. This would confuse a classifier by identifying a mutated source app as the target app. In our second obfuscation algorithm, we present an optimized scheme resulting in a trade-off between privacy and complexity overhead. For this reason, we propose a mathematical model for network obfuscation. We formulate analytically the problem of selecting the target app and the length from the target app to mutate to. Then, we propose an algorithm to solve it dynamically. Extensive evaluation of the proposed models, on real app traffic traces, shows significant obfuscation efficiency with relatively acceptable overhead. We were able to reduce a classification accuracy from 91.1% to 0.22% using the first algorithm, with 11.86% padding overhead. The same classification accuracy was reduced to 1.76% with only 0.73% overhead using the second algorithm.

CCS Concepts: • **Networks** → *Mobile and wireless security*;

Additional Key Words and Phrases: Machine learning, Internet traffic classification, side-channel information, obfuscation, anonymization, traffic padding, packet length modeling

ACM Reference format:

Louma Chaddad, Ali Chehab, Imad H. Elhadj, and Ayman Kayssi. 2021. Optimal Packet Camouflage Against Traffic Analysis. *ACM Trans. Priv. Secur.* 24, 3, Article 22 (August 2021), 23 pages.
<https://doi.org/10.1145/3442697>

1 INTRODUCTION

The explosive growth in network applications is perhaps the biggest technical phenomenon in recent years. In fact, featured applications play an integral part in our daily lives, answering those *I-want-to-know*, *I-want-to-do*, *I-want-to-go*, and *I-want-to-buy* questions. Apps are becoming the main practice of digital interaction mainly because they are simple, easy to use, and provide a multitude of new functionalities and attractive features. On the other hand, network applications can easily boost businesses by promoting them and increasing their visibility. Consumers in

Research was funded by the AUB University Research Board, and TELUS Corp., Canada.

Authors' addresses: L. Chaddad, A. Chehab, I. H. Elhadj, and A. Kayssi, Department Electrical and Computer Engineering, American University of Beirut, Beirut 1107 2020, Lebanon; emails: {lac07, chehab, ie05, ayman}@aub.edu.lb.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Association for Computing Machinery.

2471-2566/2021/08-ART22 \$15.00

<https://doi.org/10.1145/3442697>

today's business environment are on the move and they rely on network application platforms. Whether they use mobile phones, tablets, laptops, or any other mobile devices, apps have all the information they want.

According to Comscore [1], the digital population spends 69% of their media time on their smartphones, with nearly 2 hours on average spent on smartphones each day, around 45 minutes on desktops, and 24 minutes on tablets. The prevailing use of mobile platforms has been accompanied by the extensive use of apps. In 2017, users downloaded 178.1 billion mobile apps. In 2022, this number is expected to grow to 258.2 billion app downloads [2]. According to TechCrunch, the average number of apps people use is 9 apps daily and 30 apps monthly [3]. In parallel, there are currently billions of new types of devices that are getting connected through the Internet of Things (IoT) networks. The adoption of these devices is increasing at an exponential rate; therefore, new types of applications are evolving to connect our private resources to the Internet [3].

In recent years, the advent of network-based applications led to huge growth and is leading to a tremendous amount of data traffic. Researchers predict that the mobile app economy will touch 156 billion USD in 2023 [3]. Big-data scientists are processing these large amounts of network traffic data to produce broad benefits. For instance, research has proved that traffic classification is essential in network security to delineate security strategies, monitor botnet propagation, filter traffic, and the like. In fact, network intrusion detection systems can be implemented based on traffic analysis as a main element to detect suspicious network activity [4–8]. Specific usages comprise control and management of resources in TCP/IP networks through capacity planning, allocation, and network provisioning [9, 10]. Other uses include imposing precise rules on users to access the network in order to apply institutional strategies [11].

People usually download applications in line with their habits, religion, health, activities, and more. Accordingly, knowing the specific applications installed by a user can give much personal information about that individual. In order to preserve the privacy of consumers, current implementations employ different encryption techniques to preserve traffic confidentiality.

However, with the increasing popularity of machine learning techniques, the traditional means of encrypting packets are no longer feasible approaches from the privacy perspective. A passive adversary can monitor traffic patterns that remain intact after encryption, such as timing, size, direction, and count of packets in a specific network flow. Using these features, the adversary can build classifiers and detect instances of application protocols. Hence, traffic analysis is considered a big threat for the privacy of Internet users.

In this work, we aim to examine the protection of users' privacy against traffic analysis. Our goal is to guard against an eavesdropper monitoring the network and using statistical traffic analysis to infer users' sensitive data. What we need is an anonymity system that ensures strong security with acceptable computational overhead and reduced latency for interactive applications. Additionally, given that there are different types of heterogeneous apps, the obfuscation system needs to be dynamic and scalable.

In this article, we formally define the problem of privacy against traffic classification as an optimization problem in which we aim to find an ideal packet length modification that minimizes overhead. The contributions of this article are:

- Mathematical formulation of the traffic obfuscation optimization problem to defend against traffic analysis
- Analytical formulation of packet padding and chopping for various network applications
- Designing algorithms for efficient confusion of classifiers and efficient morphing with acceptable overhead
- Performing simulations and verifying the effectiveness of the algorithms using real network application traffic

The structure of this article is as follows. In Section 2, we review related work. In Section 3, we explain our scenario for attacks that we are trying to defend against. Section 4 covers the methodology of our proposed system. In Section 5, we detail our experimental classification results. Section 6 presents our conclusions and discusses future work.

2 RELATED WORK

Statistical traffic analysis has gained considerable attention from the academic and industrial research communities. It proved to be efficient in identifying security threats and providing effective management of network assets. In fact, studies proved it practical when used for application identification [14–16], anomaly detection [17, 18], user fingerprinting [19–21], and more. Using specific patterns of the packets' sizes and their timing, one can build a classifier based on machine learning techniques to extract the needed information about the network traffic.

Yet, one of the biggest concerns in information security is to provide perfect privacy. Today, network traffic analysis is considered a major threat with serious impacts on users' privacy [22]. A typical solution to these concerns is to simply encrypt networking data to hide it from a possible attacker capturing the traffic. However, this alone is not enough; ciphering does not hide all relevant information in a packetized flow. Numerous features of the encrypted network traffic, such as packet lengths, interarrival times, direction of packets, their count per flow, and many others, can still give out information about the traffic and the exact applications being used. Various studies [13, 23–25, 49] have validated that an attacker is able to disclose sensitive information about a network application user, even in the presence of strong encryption.

According to the literature, many practices can be used to obfuscate traffic and hinder the ability of an adversary to learn from network traffic. We can categorize these evasion techniques against traffic analysis into 3 groups:

- Mutation that modifies various properties of network connections [29–34, 36–38]
- Morphing that modifies one class of traffic to look like another class with respect to a given set of features [35]
- Tunneling network traffic in payload of HTTP(S) protocol to evade the path from source to destination [42]

Most past studies for traffic anonymization against traffic analysis were specific to defending against website fingerprinting [26–28, 45, 46]. In fact, one of the first attempts to secure website users from privacy attacks was introduced by Wagner and Schneier [47]. They suggest to apply random length padding on all cipher modes of the SSL protocol. Buffered Fixed-Length Obfuscator [34] is another well-known defense that sends packets at fixed sizes and times using dummy packets. However, it results in a high bandwidth overhead. More advanced techniques have been suggested in [48], in which they use camouflage to change the patterns of the data traffic on purpose by loading several web pages instead of the requested one. In [51], a traffic morphing technique (Glove) was proposed. It consists of grouping websites, whose network flows are similar, into clusters. Later, Glove uses minimum dummy traffic to shape websites in a cluster to be the same. Thus, an attacker cannot classify a specific website but only the cluster to which it fits. The authors in [50] present the WeFDE technique to perform information leakage on a dataset. This technique estimates the bits of information that are revealed by particular features of a website's traffic. Recent studies [44] have shown that the applicability of those paradigms is sometimes limited to websites only. According to [43], website-based activities differ from in-app user activities. Consequently, the solution for obfuscating the traffic could also be different. Therefore, only a few solutions of the previous works in security for obfuscation, using machine learning practices, can be used to thwart side-channel information leakage specific to network applications.

To minimize the amount of information leaked, some works suggest features quantization, such as padding packets to fixed sizes [30, 31]. Other defenses [29] produce a random “noise,” such as added dummy packets to mask users’ activities. These techniques for traffic anonymization are currently dominant. Their effectiveness is acceptable; however, the large added overhead is not practical since it severely worsens the efficiency and performance of the original network protocols. Padding encrypted packet sizes to their maximum transmission unit (MTU), for example, could end up more than doubling the amount of data sent [32]. Also, these approaches lead to undesirable delay and bandwidth waste [33]. Accordingly, such unnecessary padding is not a reasonable solution to the problem. Moreover, it has been shown that this mechanism can be countered to reveal users’ activities [34].

In [35], Wright et al. proposed traffic morphing as a general type of defense against network fingerprinting attacks, relying on convex optimization. However, there is an issue with the practicality of this method given the needed computations; hence, it cannot be used in devices with restricted resources. Also, it increases latency due to the generation of random numbers for each input packet.

In [36], the authors aim to remove the correlation between the original and the padded packets by optimally hiding the actual packet lengths. The main difference between [36] and our method is the effectiveness of the methods given that [36] relies only on padding and not fragmentation. In [37], the same authors proposed traffic masking by taking into consideration several traffic features with minimum overhead.

In [38], Apthorpe et al. describe a stochastic traffic padding framework to protect smart homes against traffic analysis. Their method consists of shaping traffic to a fixed traffic pattern, in intermittent periods, to limit the information exposed about users’ activities through traffic analysis. This technique obfuscates traffic originating from IoT devices. However, it could result in network latencies. In addition, this method is not scalable to other types of networking traffic because it is a primarily different problem dealing with analysis of user traffic patterns.

In our previous work [13], we morphed packet lengths of a source app to packet lengths of another target app using padding and fragmentation. We confused a mobile application traffic by shaping an app’s smallest packet length in a flow to the smallest packet length of another app’s flow. Using this method, we were able to reduce a classifier’s accuracy from 91.1% to 15.7% with 34.6% overhead. In [39], we presented a generalized method that has 3 variations of [13] and included mutating lengths or/and interarrival times of packets. We were able to reduce a classifier’s accuracy from 91.1% to 7% with only 12% overhead. In [40], we applied anonymization to the size probability distribution characteristic of an app’s packet traffic for the first time. We regenerated an app’s packet sizes from its best-fitted distribution model. Then, we mutated packet lengths of a source app that we were trying to protect to regenerated packet sizes of another target app. This method proved to reduce a 91.1% classification accuracy to 0.9% with only 12.51% overhead. While our earlier obfuscation algorithms resulted in significant results, still there has not been a formal definition of our security problem. No mathematical solution has been assessed yet to optimally balance both efficiency and performance for a solid obfuscation algorithm.

In brief, despite the efforts to obfuscate side-channel information in recent years, it is still possible to determine networking traffic flows. In [34], Dyer et al. reviewed techniques known in the literature to defend against traffic analysis and revealed their key weaknesses. Many works have been presented as promising to reduce the performance of a traffic classifier but their impact on the actual performance degradation has been ignored. To the best of our knowledge, there is no formal statement of the traffic anonymization problem. Most suggested solutions were specific to certain types of traffic, such as mobile applications, websites, IoT devices, and so forth. Also, no

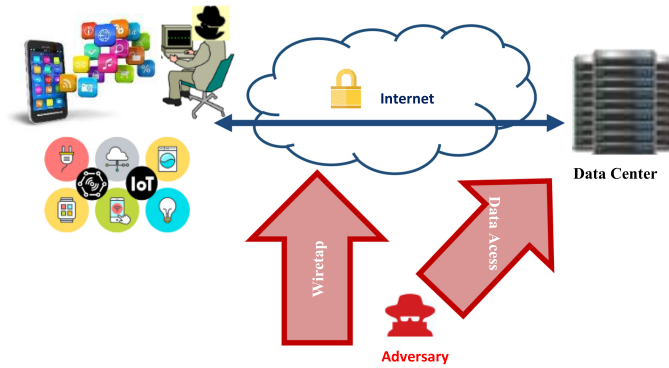


Fig. 1. Threat Model.

efficient optimal solution to minimize the overhead cost and implementation complexity to defeat classifiers has been suggested yet.

3 ATTACK MODEL

We study the network traffic, in which we identify 2 hosts running an application. Despite using an encrypted channel for communication, there is still data leakage to an attacker monitoring the flows between the two hosts. Traffic analysis attacks can be defined in the form of classifiers that try to recognize the application within the encrypted connection.

Figure 1 illustrates our threat model, in which an adversary aims to determine a victim's online activity. We consider users of network-based applications where the program used, or the data related, or both exist on a network. There is a wide range of network applications, such as chatting, downloading, uploading, online gaming, video streaming, webpage browsing, voice-over-IP, and so on. Users usually perform these activities by running particular applications or service apps in a mobile phone, applications on devices in an IoT network, webpages, computer applications, and more. We assume that the attacker, employing network traffic fingerprinting, is able to sniff the encrypted exchanged data between the node and the server. The adversary uses sniffer software (e.g., Wireshark) and does not have any knowledge about the software or encryption schemes implemented by the user's apps. Even worse, the adversary can have physical access to servers in the data center where data is stored and ultimately get traces of the encrypted traffic. We assume that the adversary cannot decrypt the packets and wants to reveal information about the network users and the applications they use. To this purpose, our adversary uses machine learning to create traffic fingerprints and to match the precise application the user is running, even though packet payloads are not readable.

In this context, the attacker inspects side-channel information (IP packet headers and metadata) that leak from the ciphered app traffic. The attacker constructs a classifier for a number of network applications and then matches the recorded traffic of the user to infer the exact app being used. This attack is passive and unnoticeable by users. Our objective is to provide sufficient protection to thwart such data leaks. We aim to prevent wiretapping and malicious identification of encrypted Internet traffic through traffic analysis.

Figure 2 shows an example of a traffic classification attack. Using this, we test our proposed obfuscation model in Section 5. We consider a mobile phone with which the consumer uses apps connected to the Internet. The traffic created by the apps is end-to-end encrypted. The adversary eavesdrops on the traffic, records the encrypted traces of the user and tries to learn information about the user by detecting the apps being used.

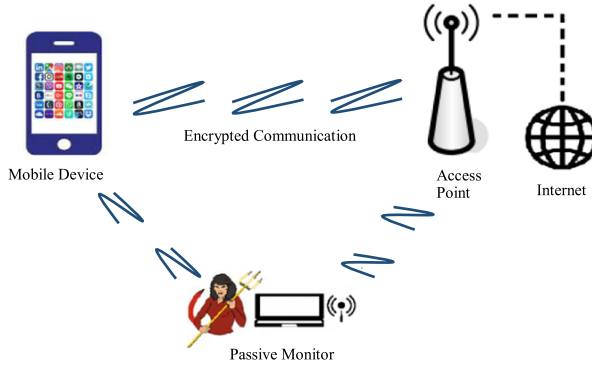


Fig. 2. Mobile Traffic Classification Attack.

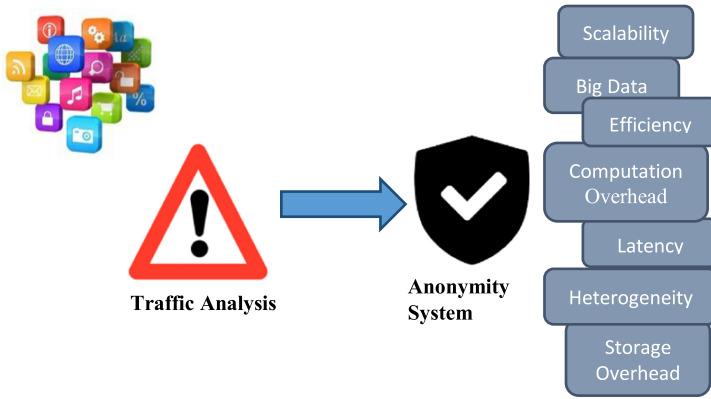


Fig. 3. Problem Definition.

To this end, the attacker implements supervised machine learning algorithms with a training and testing phase. During training, the algorithms are given (X_i, y_i) where each X_i is a vector of features and y_i is a label. During testing, the classifier is given a vector Z and must return a label for the detected class indicating the precise mobile app that created the flow. In our situation, a vector X_i has information about the lengths, interarrival times, and direction of packets in the encrypted flow generated by an application.

In summary, current implementations of network-based applications offer limited security assurances against traffic analysis. What we need is an obfuscation system that ensures strong security. The proposed solution needs to be efficient in the first place, to have low computational and storage overhead, and to have reduced latency on the traffic. Additionally, it needs to be dynamic, scalable, and able to deal with big data. Figure 3 depicts our formal statement of the problem.

4 PROPOSED FRAMEWORKS

We aim to mitigate network fingerprinting attacks based on statistical traffic analysis by confusing an app’s traffic. As suggested by the literature, the main characteristics of encrypted traffic are packet lengths and timing delay of consecutive packets. Our models will focus on mutating the packet size feature because changing interarrival time would impose delays and affect performance and, thus, is not practical, especially for the dynamic type of applications.

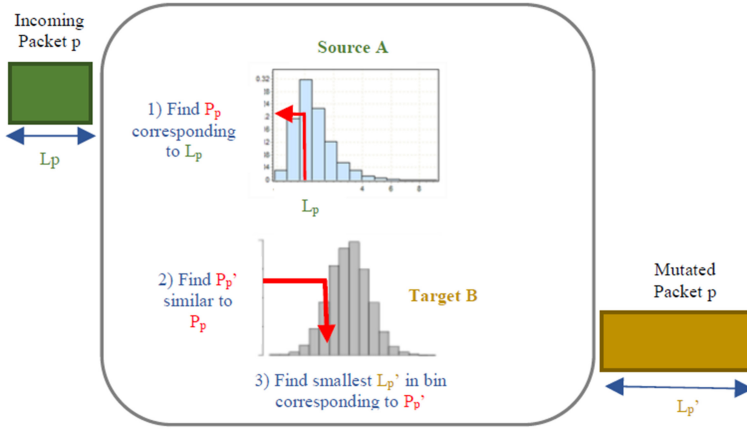


Fig. 4. Algorithm 1.

In what follows, we discuss first our preliminary technique. We then present our extended algorithm, which ensures optimization of privacy and performance.

In practice, our mutation algorithms act as a proxy between the applications and the network stack. In real-time network applications, we set flags in the packet header to indicate whether the packet is padded or fragmented. The packet also has a field containing the number of padding bytes that would be omitted by the receiving endpoint to recover the original packet.

4.1 First Model

In our first model (Algorithm 1), we modify each packet in the flows, created by the source app to defend, such that their distribution becomes similar to packet sizes of a different target app. This entails resampling packet sizes from probability distribution of a target app traffic that is dissimilar from the actual packet lengths created by the source app. In other words, we split and pad network packets so that the altered flow looks like a predefined target distribution.

Figure 4 illustrates our traffic mutation technique, labelled as Algorithm 1. Given a pair of applications A and B, where A is the source app to defend and B is the target app that the source app should resemble, we choose a bin size w for packet sizes. We derive the probability distribution P_A and P_B with respect to bins of packet lengths of A and B, respectively.

For each incoming packet p of A of size L_p having a probability P_p , we find the most similar probability $P_{p'}$ in target app B. Then, and to minimize overhead, we find the smallest size $L_{p'}$ among the sizes in the bin of probability $P_{p'}$ in B.

To find the most similar $P_{p'}$ to P_p , we consider the vector P_B of probabilities for the different bins of packet sizes of the target app. We find $P_{p'}$ among the values in vector P_B satisfying (1).

$$\text{Minimum } |P_p - P_{p'}| \quad (1)$$

Then, we compare $L_{p'}$ and L_p . If $L_{p'} > L_p$, we pad the packet from A with zeroes such that the resulting packet size is $L_{p'}$ and we send the padded packet. Otherwise, the algorithm splits the data into several packets. We send L_i bytes of the original packet and continue sampling from P_B until all bytes of A are sent. Packet sizes that are output onto the network appear to be drawn from the target application distribution.

Using Algorithm 1, we essentially mutate the lowest probabilities of packet lengths of A to the lowest probabilities of packet lengths of B and the biggest ones from A to the biggest in B. As long as the source process generates a sufficiently large number of packets from a source distribution

A, the output will converge in distribution to that of the target B, realizing morphing of A into B. Therefore, the classifier confuses an incoming packet from A of a size specific to B and classifies it in the target app class. This technique realizes morphing but it can cause a huge overhead in terms of padding depending on the choice of the target app for each source app.

One can argue that in spite of applying this technique, a user's behavioral patterns still leak. To obfuscate their behavior, users may choose the target app that they do not usually use. Many apps can be appropriate in this case. Users can choose the target app that is the most dissimilar to their normal behavior for better obfuscation.

Also, it would be beneficial to change the target app at each run to prevent the attacker from knowing the mapping of the source app to the target app.

4.2 Second Model

To overcome the overhead issue, we propose our second model (Algorithm 2), in which we formulate an optimization problem that yields a constructive solution for an algorithm that achieves maximum obfuscation with minimum padding. To this end, we consider one source app to protect and a number of N target apps to mutate to. We find the probability distribution with respect to packet lengths of the source app and those of the target apps traffic.

4.2.1 Problem Statement. At a high level, our optimization problem is defined as follows. Given a source application, our goal is to mutate its traffic such that it resembles and is confused as some target application (i.e., a different application) with respect to a given set of features. The user first chooses the source application to defend as well as a set of target applications to make the source application look like. Of course, we need to ensure that the implemented mutation is the most effective, for some measure of efficiency, such as the number of overhead bytes added. The user applies the optimization technique that dictates how each packet size from the source application should be mutated such that the resulting distribution resembles the target with a minimum of overhead. Naturally, after the mapping of the packet sizes is performed offline, the mutation algorithm captures the data to be conducted across the network before headers are calculated or encryption is applied. The mutated traffic is then sent to the network stack, encrypted, and sent across the network as usual.

Our objective is to maximize network traffic anonymization (i.e., minimize the accuracy of a classifier detecting certain applications) and to minimize the overhead resulting from padding. This would be realized by means of (1) selecting the optimum target app and (2) determining the optimum packet size to mutate to. As such, the mutation algorithm can be formulated as an optimization problem to mutate a source application into a target application.

An application i can be defined by $[m_i, L_i, P_i]$:

- The number of possible packet size ranges of width w is denoted by m_i .
- The vector $L_i = [L_i^1, L_i^2, \dots, L_i^{m_i}]$ of smallest lengths in each bin of size w . For example, $L_i^{m_i}$ is the smallest length in the last bin of application i and having a total number m_i of bins. The average of L_i is denoted by $Avg(L_i)$.
- The probability distribution characteristic denoted by $P_i = [P_i^1, P_i^2, \dots, P_i^{m_i}]$ represents the vector of length probabilities for bins of size w . For example, $P_i^{m_i}$ is the probability of length $L_i^{m_i}$ for application i .

An incoming packet from the source application can be defined by its size l and the probability p of the bin to which l belongs. We define a binary variable for each application that indicates whether the app is selected as target app or not; x_i denotes the variable for the i^{th} app. Given a set of N applications, the vector X (of length N) represents $[x_1, x_2, \dots, x_N]$.

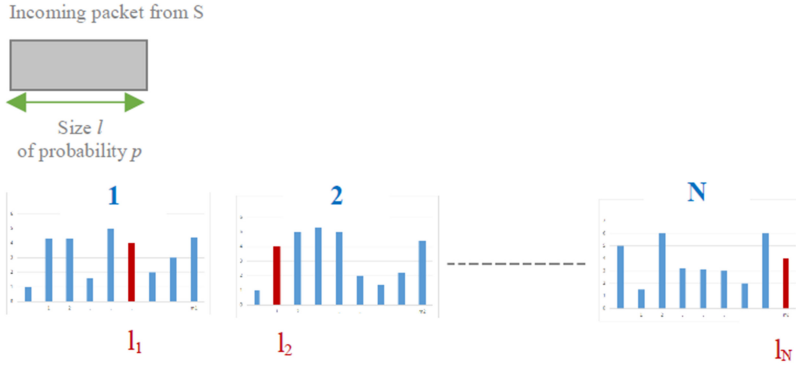


Fig. 5. Algorithm 2 Camouflage Scheme.

Also, we delineate a binary variable for each bin of an application i that indicates whether the packet bin is selected to be mutated to or not. It is worth noting that the smallest size in a bin is used for mutation. y_i^j denotes the variable for the j^{th} bin of application i . Given a set of m_i ranges, the vector y_i (of length m_i) designates $[y_i^1, y_i^2, \dots, y_i^{m_i}]$.

In order to minimize the accuracy of the classifier, we need to find the probability of a packet length in a bin of a target app such that it is similar to p . At the same time, to minimize the overhead, the selected packet length must be similar to l . In case the difference between p and the designated probability is too small (less than 0.1 in our case), the difference between the selected length and l needs to be checked to be less than the average difference. This would guarantee that the overhead does not reach a large value.

The obfuscation optimization problem for mutating an incoming packet from a source app is as follows:

Find the vector X (of length N), and the vector y_i (of length m_i), and *minimize*:

$$\text{Min} \sum_{i=1}^N x_i \left(\sqrt{\sum_{j=1}^{m_i} y_i^j (|P_i^j - p|^2 \times |L_i^j - l|^2)} \right)$$

Subject to:

- (1) $P_i^j \in [0, 1]; \quad i = 1, \dots, N$
- (2) $|L_i^j - l| < |Avg(L_i) - l|$ for $|P_i^j - p| < 0.1$
- (3) $\sum_{i=1}^N x_i = 1; \quad x_i = 0 \text{ or } 1; \quad i = 1, \dots, N$
- (4) $\forall i \in [1, N], \sum_{j=1}^{m_i} y_i^j = 1; \quad y_i^j = 0 \text{ or } 1; \quad j = 1, \dots, m_i$

4.2.2 Solution to the Optimization Problem. The ultimate mutation solution for our optimization problem is to identify for each length probability from the source app the most similar ones from the target apps. Then, we mutate the length from the source app to the smallest corresponding length to these similar probabilities from the target apps.

Figure 5 depicts our solution. We consider a source application S to protect and a set of target applications to mutate to, numbered from 1 to N . For each incoming packet of S of size l having a probability p , we find the bins of packets within the N applications with similar size probabilities $[p_1, p_2, \dots, p_N]$ using (1). We mark these bins in red in Figure 5. Then, we find their corresponding

Table 1. Experiment Dataset

#	App	Type	packets/class	% Packets	# Flows
1	Skype	Video Call	197,115	0.350	520
2	Facebook	Interactive browsing	122,422	0.218	2,560
3	8 ball pool	Game	121,663	0.217	2,064
4	WhatsApp	Messaging	3,315	0.006	435
5	Viber	VOIP	82,662	0.147	475
6	YouTube	Video streaming	35,027	0.062	813

First, we divide the traces in each app traffic set into bins of size w of 50 bytes. We study the probability of each bin of packet lengths across all instances. Graphs in Figure 6 display the statistical analysis results of packet size profiling for the 6 apps of our dataset.

smallest lengths $[l_1, l_2, \dots, l_N]$ in the bin they belong to. We choose to mutate l to the smallest length among $[l_1, l_2, \dots, l_N]$.

The classifier would notice specific lengths from the N target apps. Then, it would classify the incoming packets in their corresponding target app rather than random classification. The use of optimization ensures minimum padding overhead when mutating the source packets.

Additionally, the generation of the matrices mapping each bin of a source packet to a size in another target app can be done offline. Therefore, the whole method becomes more practical, resulting in minor latency, mainly for the dynamic type of applications.

By applying our second algorithm, the user's behavior is obfuscated by design because each single source app is mutated to several target apps at once. This would cause confusion for the classifier, and the attacker would not be able to discriminate a single app efficiently. Hence, applying our second algorithm would increase the unpredictability of the source as well as the target app and, ultimately, the user's behavior.

5 EXPERIMENTAL EVALUATION

In what follows, we empirically evaluate the efficiency of our system, in which we explain our classification experiment and elaborate on our evasion defense methods. First, we study the packet size probability distribution for the used apps. Next, we define the traffic classifiers where machine learning techniques are used to promote a classification attack. Lastly, we present a complete evaluation of our solution models using the different combinations of app traffic in our dataset. We assess our 2 suggested evasion approaches by studying their efficiency and comparing them to 2 other algorithms from the literature "MTU quantization" and "Random Padding." We also provide a practical and adaptive way for our classifiers to admit or reject traffic flow to a certain class label using a similarity metric based on root-mean-square deviation.

5.1 Mobile Apps Packet Size Probability

In our experiments, we utilize the dataset from [12], which consists of traces of network traffic for 6 commonly used apps collected on Android smartphones and manually classified. Each packet in the dataset is defined by a set of attributes: Packet Number, Timestamp, Packet Length, Interarrival Time, Direction (IP source/IP destination), and App Label. For each of the 6 apps, we extract traffic flows for a fixed time period of 1 s between two IPs with the same protocol and ports. We summarize the types of the 6 apps, the size of each class, their sample proportions, and the count of flows extracted in Table 1.

Table 2. Dataset Accuracy

Model	Parameters	Accuracy
SVM	Quadratic Kernel	76.7%
Bagged Trees	Minimum Leaf Size = 2	90.0%
KNN	Number of neighbors = 5; Distance Weight = Squared inverse	83.9%
Random Forest	Distance Weight = Squared inverse	91.1%

Table 3. Results of Skype Mutation Using Algorithm 1

Algorithm	SVM	BT	KNN	RF	Overhead
Original % Accuracy	76.7	90	83.9	91.1	—
% Accuracy of Skype mutated to Facebook	47.31	53.08	40.00	51.2	8.24%
% Accuracy of Skype mutated to Game	47.88	54.62	42.88	53.87	15.7%
% Accuracy of Skype mutated to WhatsApp	47.88	54.52	42.88	54.1	20.48%
% Accuracy of Skype mutated to Viber	49.62	58.46	45	57.98	2.64%
% Accuracy of Skype mutated to YouTube	47.31	53.08	40	52.55	49.85%

5.2 Classification Attack Experiment and Results

To assess our obfuscation methods, we build 4 machine learning algorithms as explained in our previous work [13]. We choose to implement 27 features based on Wrapper methods along with analysis of dataset properties and relationships among features [12]. The feature-extraction phase generates an imbalanced dataset in which the classes are not similarly represented. Consequently, we up-sample smaller classes to balance the data and get a total of 11,970 instances to build the classifiers. After that, we divide the data into 75% for the training and 25% for testing. We use the “one versus one” scheme for the multi-classification problem and we evaluate our models using 5 folds’ cross-validation. The accuracy of the different classifiers and their corresponding parameters are presented in Table 2.

The four app network classifiers have a considerable classification accuracy. Therefore, the security of a network application user is threatened. In our next experiment, we use these four classifiers to evaluate our obfuscation algorithms.

5.3 Obfuscation Experiments and Results

Here, we present two experiments evaluating the two different anonymization models of mutating packet lengths, as described in Section 4. We investigate the reduction in accuracy caused by these 2 mutation methods. The main difference between the 2 models is the number of target apps to which we mutate. As the number of target apps increases, the overhead resulting from padding decreases. This algorithm would result in better attack evasion results but the morphing becomes less significant. One of these models can be selected based on the preference of the user for morphing defense or confusion defense with minimum overhead.

In this first experiment (Algorithm 1), we choose the source app to confuse and the target app to resemble. We save the probability of each packet length bin for the source and target apps. Then, for each incoming packet from the source app, we find the probability of its corresponding bin size. We find the most similar length probability in the target app and its corresponding bin. We pad or chop the incoming packet length to the smallest length inside the selected target bin.

Using our 4 model classifiers, we predict the classes of the modified traffic. We present, in Tables 3 to 8, comprehensive simulation results of mutation using Algorithm 1 of each

Table 4. Results of Facebook Mutation Using Algorithm 1

Algorithm	SVM	BT	KNN	RF	Overhead
Original % Accuracy	76.7	90	83.9	91.1	—
% Accuracy of Facebook mutated to Skype	44.3	58.95	47.81	57.4	85.21%
% Accuracy of Facebook mutated to Game	45.59	59.1	47.89	58.23	32.02%
% Accuracy of Facebook mutated to WhatsApp	44.57	58.09	47.31	57.5	27.7%
% Accuracy of Facebook mutated to Viber	45.43	58.01	49.96	56.2	10.51%
% Accuracy of Facebook mutated to YouTube	44.45	58.48	47.5	58.9	19.2%

Table 5. Results of Game Mutation Using Algorithm 1

Algorithm	SVM	BT	KNN	RF	Overhead
Original % Accuracy	76.7	90	83.9	91.1	—
% Accuracy of Game mutated to Skype	0.39	0.34	2.23	0.56	80%
% Accuracy of Game mutated to Facebook	0.19	0.24	2.13	0.44	21.37%
% Accuracy of Game mutated to WhatsApp	0.24	0.29	2.33	0.24	71%
% Accuracy of Game mutated to Viber	0.48	0.19	2.18	0.22	11.86%
% Accuracy of Game mutated to YouTube	0.53	0.15	1.74	0.18	36.33%

Table 6. Results of WhatsApp Mutation Using Algorithm 1

Algorithm	SVM	BT	KNN	RF	Overhead
Original % Accuracy	76.7	90	83.9	91.1	—
% Accuracy of WhatsApp mutated to Skype	5.52	14.02	11.3	13.4	70.76%
% Accuracy of WhatsApp mutated to Facebook	7.13	14.48	14.25	13.26	25.11%
% Accuracy of WhatsApp mutated to Game	5.06	14.02	10.34	14.01	21.73%
% Accuracy of WhatsApp mutated to Viber	3.22	14.94	11.95	13.29	7.76%
% Accuracy of WhatsApp mutated to YouTube	5.29	15.4	13.33	14.32	39.89%

Table 7. Results of Viber Mutation Using Algorithm 1

Algorithm	SVM	BT	KNN	RF	Overhead
Original % Accuracy	76.7	90	83.9	91.1	—
% Accuracy of Viber mutated to Skype	5.26	33.26	9.68	32.42	94.92%
% Accuracy of Viber mutated to Facebook	7.79	33.68	6.95	32.22	70.43%
% Accuracy of Viber mutated to Game	9.68	34.73	11.79	34.3	49.88%
% Accuracy of Viber mutated to WhatsApp	8	34.74	8.84	33.29	76.85%
% Accuracy of Viber mutated to YouTube	9.89	32.84	10.53	33.1	89.81%

application in our dataset to the 5 other different apps, correspondingly. Mutating Skype and Facebook was not as effective as mutating the other apps. For the sake of simplicity, we will explain in what follows the inefficiency of Skype mutation and the efficiency of Game mutation. The other apps' results can be explained likewise.

As argued before, Algorithm 1 would mutate probabilities of a source app to similar ones in the target app. Thus, based on Figure 6, mutating Skype into any of the other apps would result in mutating all bins of Skype into a single bin from the other target apps. Also, this single bin happens to be a minority in comparison with the other bin probabilities of the target app. For example,

Table 8. Results of Viber Mutation Using Algorithm 1

Algorithm	SVM	BT	KNN	RF	Overhead
Original % Accuracy	76.7	90	83.9	91.1	—
% Accuracy of YouTube mutated to Skype	9.86	8.61	17.22	10.1	27.48%
% Accuracy of YouTube mutated to Facebook	9.95	8.61	17.22	9.93	85.34%
% Accuracy of YouTube mutated to Game	9.76	8.61	17.22	10.4	5.69%
% Accuracy of YouTube mutated to WhatsApp	9.84	11.81	19.19	9.72	8.3%
% Accuracy of YouTube mutated to Viber	9.96	11.32	20.54	9.82	1.68%

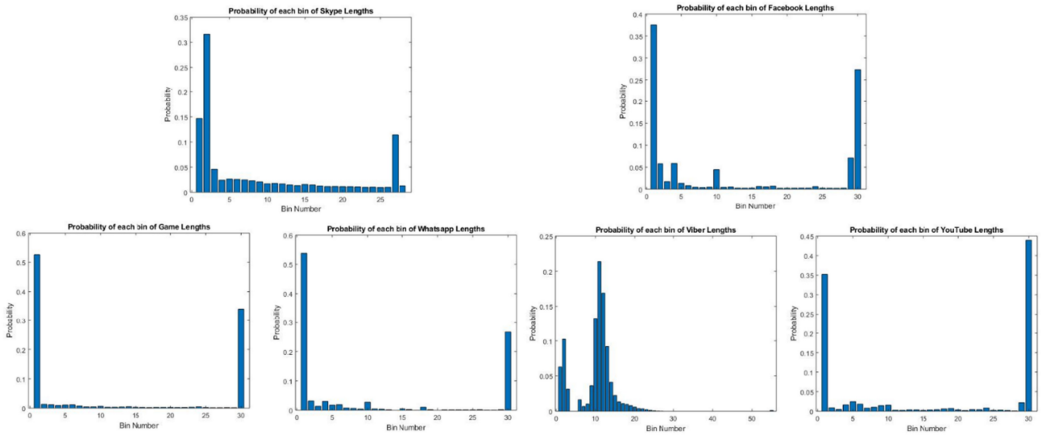


Fig. 6. App Traffic Packet Size Distribution.

when mutating Skype to Game, first bins of Skype with a probability of 0.32 would be mutated to bin number 30 of Game having a similar probability of 0.35. Also, the last bins of Skype with a probability of 0.13 would mutate to bin number 30 in Game. Therefore, mutating Skype would converge into mutating most bins into a single bin from the target app that is not representative of the target app. Thus, the new reformed traffic would neither fit into Skype probability distribution nor would it fit into the target probability distribution. This would not totally confuse the classifier to classify randomly.

In the case of Game mutated to Skype, first bins of Game are mutated to bin number 2 of Skype, and the last bins of Game are also mutated to bin number 2 of Skype because it is the most similar one. However, bin number 2 is representative for Skype because it has the highest probability of 0.32. Hence, the overall probability distribution of mutated Game was altered and the classifier identifying Game would confuse it as Skype. The same explanation clarifies the efficiency of Game mutated to Facebook and Game mutated to Viber.

When mutating Game to WhatsApp, the first bins of Game are mutated to the first bins of WhatsApp and the last bins of Game to the last bins of WhatsApp. Thus, the probability distribution of mutated Game would fit into the probability distribution of WhatsApp, confusing the classifier to classify the mutated Game traffic as WhatsApp.

When applying Game to YouTube using Algorithm 1, the first bins of Game are mutated to the last bins of Viber and the last bins of Game to the first bins of Viber. Thus, the probability distribution of mutated source traffic is totally changed into that of the target app. Therefore, the classifier would classify the mutated Game traffic as YouTube.

Table 9. Results of Viber Mutation using Algorithm 1

	Skype	Fb	Game	WhatsApp	Viber	YouTube
Game (%)	13.87	12.94	0.22	1.73	69.23	2.01

Table 10. Optimization Results of Skype Source App Using Algorithm 2

Source Bin #	Optimum Target App	Optimum L to Mutate to
1	Game	104
2	YouTube	54
3	Game	104
4	Game	104
5	Game	104
6	Game	104
7	Game	104
8	Game	104
9	Game	104
10	Viber	100
11	Viber	100
12	Viber	100
13	Game	104
14	Game	104
15	Viber	100
16	Game	104
17	Game	104
18	Game	154
19	Game	154
20	Viber	210
21	Viber	210
22	Viber	210
23	Viber	120
24	Viber	120
25	Game	204
26	Viber	220
27	Viber	62
28	Game	104

The simulation results using Algorithm 1 are shown in Table 5. For example, mutating Game to Viber can decrease SVM classifier’s accuracy from 76.7% to 0.48%, Bagged Trees classifier’s accuracy from 90% to 0.19%, KNN classifier’s accuracy from 83.9% to 2.18%, and Random Forest classifier’s accuracy from 91.1% to 0.22% with only 11.86% average overhead. According to Table 9, 69.23% of the Game traffic was confused as Viber traffic. This demonstrates the efficiency of our method to morph one class of Game traffic into the Viber class.

In the second experiment (Algorithm 2), we choose the source app to mutate. For each incoming packet from the source app, we implement the solution of the optimization problem, described in Section 4.2, to select the optimal target app and packet size to mutate to. Tables 10 to 15 present, for each bin size of a source app from our dataset, the corresponding optimum target app and optimum packet size to mutate to using Algorithm 2.

Table 11. Optimization Results of Facebook Source App Using Algorithm 2

Source Bin #	Optimum Target App	Optimum L to Mutate to
1	YouTube	54
2	Viber	50
3	Viber	100
4	Viber	50
5	Game	104
6	Viber	230
7	YouTube	154
8	Viber	250
9	YouTube	154
10	Game	104
11	YouTube	154
12	Viber	240
13	Viber	260
14	Viber	270
15	Viber	260
16	Viber	240
17	Viber	240
18	Viber	110
19	Viber	270
20	Viber	270
21	Viber	270
22	Viber	260
23	Viber	270
24	Viber	240
25	Viber	270
26	Viber	270
27	Viber	270
28	Viber	260
29	Viber	50
30	YouTube	54

Table 16 presents mutation results using Algorithm 2 of each app in our dataset. For example, mutating Game using the second algorithm can decrease SVM classifier’s accuracy from 76.7% to 0.24%, Bagged Trees classifier’s accuracy from 90% to 0.001%, KNN classifier’s accuracy from 83.9% to 1.89%, and Random Forest classifier’s accuracy from 91.1% to 1.76% with only 0.73% average overhead.

We also assess our results in terms of the F1 score. The highest possible value of F1 score is 1, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero. Table 17 presents F1 score results using Algorithm 2 of each app in our dataset.

For further evaluation, we suggest a simple metric “scalar confidence” that we denote by Sc to determine the similarity match of an incoming traffic to one of those in our dataset. Given 2 samples of traffic, A and B, $Sc(A||B)$ is the root-mean-square deviation between the probability distribution of A and that of B. In our case, A would be the probability distribution of an incoming traffic that we would like to determine to which app of our dataset is most similar. Thus, $Sc(A||B)$ would measure how well an incoming sample traffic matches a certain app of our dataset. Hence,

Table 12. Optimization Results of Game Source App Using Algorithm 2

Source Bin #	Optimum Target App	Optimum L to Mutate to
1	Facebook	54
2	WhatsApp	154
3	Viber	210
4	YouTube	104
5	Viber	120
6	Viber	210
7	Viber	110
8	YouTube	154
9	YouTube	154
10	Viber	240
11	Viber	260
12	Viber	260
13	Viber	250
14	YouTube	154
15	Viber	260
16	Viber	260
17	Viber	270
18	Viber	260
19	Viber	260
20	Viber	260
21	Viber	270
22	Viber	270
23	Viber	260
24	YouTube	154
25	Viber	270
26	Viber	270
27	Viber	280
28	Viber	270
29	Viber	280
30	Facebook	54

it would determine to which extent a certain obfuscation algorithm can modify the probability distribution of side information of an app traffic. Given the set Q of length probability distributions of an app traffic dataset, and q a length probability distribution of an incoming unknown traffic that we would like to determine, q is classifiable as t with respect to Q , if

$$\forall t, t' \in Q, Sc(q||t) < Sc(q||t') \text{ and } Sc(q||t) \leq c, \text{ where } c \text{ is the threshold for } Sc.$$

In our experiments, we choose $c = 0.01$. We demonstrate in Table 18 scalar confidence values for our 4 classifiers before and after applying Algorithm 2. High values of Sc after applying our obfuscation algorithm indicate its success in fooling the adversary. This is because the similarity matching between the mutated traffic and apps of our dataset is very low.

Also, we compare our schemes to 2 known obfuscation techniques from the literature, MTU quantization and Random Padding, and we use the same dataset as a testbed. MTU quantization pads the packets to their maximum transmit unit (MTU). Random Padding pads each packet with a random number of packets. The number of padding bytes follows a uniform probability

Table 13. Optimization Results of WhatsApp Source App Using Algorithm 2

Source Bin #	Optimum Target App	Optimum L to Mutate to
1	Facebook	54
2	Viber	73
3	Game	104
4	Viber	73
5	Viber	100
6	Viber	100
7	Viber	230
8	Viber	240
9	Viber	250
10	Viber	73
11	Viber	250
12	Viber	260
13	Viber	280
14	YouTube	154
15	Viber	270
16	Viber	300
17	Viber	120
18	Viber	270
19	Viber	592
20	Viber	592
21	Viber	592
22	Viber	592
23	Viber	592
24	Viber	270
25	Viber	290
26	Viber	290
27	Viber	270
28	YouTube	54

distribution from the current size to MTU in order to maximize randomness. We run the same experiment on MTU quantization and Random Padding schemes to classify the classes of 6 mutated apps in Table 19. The MTU quantization scheme reduces SVM classifier's accuracy from 76.7% to 31.8%, Bagged Trees classifier's accuracy from 90% to 22.3%, KNN's classifier from 83.9% to 25.4%, and Random Forest classifier's accuracy from 91.1% to 20.3%. On the other hand, the Random Padding algorithm reduces SVM classifier's accuracy from 76.7% to 52.3%, Bagged Trees classifier's accuracy from 90% to 41.9%, KNN's classifier from 83.9% to 43.6%, and Random Forest classifier's accuracy from 91.1% to 40.4%. Our algorithms realize better results in terms of dropping the classifiers' accuracy. Also, MTU quantization and Random Padding result in an overhead of 223.5% and 84.2%, respectively, which is ineffective. Padding packets randomly or to their maximum sizes ruins the network performance because extreme padding carries a large volume of data to send, making the MTU and Random Padding methods not practical.

According to the results in this section, our countermeasures optimally balance performance and security. They are more suitable for network devices with restricted plans mainly because they decrease overhead in terms of data and bandwidth. Also, they are appropriate for constrained

Table 14. Optimization Results of Viber Source App Using Algorithm 2

Source Bin #	Optimum Target App	Optimum L to Mutate to
1	Game	104
2	Game	104
3	Game	104
4	WhatsApp	884
5	Game	104
6	WhatsApp	408
7	Game	254
8	Game	104
9	Skype	53
10	Skype	53
11	Skype	53
12	Game	104
13	Game	104
14	Game	104
15	Game	104
16	Game	254
17	Game	204
18	Facebook	304
19	WhatsApp	408
20	Facebook	404
21	Game	554
22	WhatsApp	809
23	WhatsApp	1060
24	Skype	1253
25	WhatsApp	884
26	WhatsApp	884
27	WhatsApp	884
28	WhatsApp	884
29	WhatsApp	1060

networking equipment and they do not necessitate much processing and computations. In addition, our algorithms are highly scalable to large datasets. Furthermore, another advantage of our schemes is that the matrices mapping the mutation needed to modify source app into target app can be realized offline. Thus, our methods can run in a dynamic online manner and the whole process would induce very low latency.

6 CONCLUSION

Network security is a challenging concern and a key requirement for network designers. With network applications, security implications are well pronounced since the impact of attacks is drastic. This is mainly because network applications are central to our life, including a wide variety of applications, from simple location awareness to critical health care uses. Current implementations of network applications still reveal traffic features out of encrypted traffic. These features can be exploited by machine learning techniques to obtain leakage on the information carried,

Table 15. Optimization Results of YouTube Source App Using Algorithm 2

Source Bin #	Optimum Target App	Optimum L to Mutate to
1	Facebook	54
2	Game	204
3	Viber	250
4	Viber	100
5	Game	104
6	Viber	100
7	Viber	230
8	Viber	120
9	Game	104
10	Viber	100
11	Viber	270
12	Viber	270
13	Viber	260
14	Viber	260
15	Viber	270
16	Viber	260
17	Viber	250
18	Viber	240
19	Viber	110
20	Viber	250
21	Viber	270
22	Viber	250
23	Viber	250
24	Game	204
25	Viber	270
26	Viber	260
27	Viber	270
28	Viber	290
29	Game	104
30	Facebook	54

Table 16. Results of Apps Mutation Using Algorithm 2

Algorithm	SVM	BT	KNN	RF	Overhead
Original % Accuracy	76.7	90	83.9	91.1	—
% Accuracy of Skype Mutated	41.93	53.48	42.68	41.24	4.28%
% Accuracy of Fb Mutated	46.31	70.21	48.57	45.92	7.91%
% Accuracy of Game Mutated	0.24	0.001	1.89	1.76	0.73%
% Accuracy of WhatsApp Mutated	4.72	13.53	12.62	11.91	0.106%
% Accuracy of Viber Mutated	13.12	32.05	12.61	13.19	0.57%
% Accuracy of YouTube Mutated	15.26	6.09	16.77	14.99	0.93%

Table 17. F1 Score of Apps Mutation Using Algorithm 2

Algorithm	SVM	BT	KNN	RF
Skype Mutated	0.42	0.56	0.41	0.45
Fb Mutated	0.41	0.68	0.52	0.44
Game Mutated	0.001	0	0.001	0.012
WhatsApp Mutated	0.03	0.01	0.02	0.02
Viber Mutated	0.02	0.03	0.01	0.01
YouTube Mutated	0.01	0.05	0.01	0.04

Table 18. Scalar Confidence (Sc) of Apps Mutation Using Algorithm 2

Algorithm	SVM	BT	KNN	RF
Original Traffic	0.0086	0.0037	0.0059	0.0029
Skype Mutated	0.0352	0.0299	0.0365	0.0337
Fb Mutated	0.0343	0.0218	0.0341	0.0384
Game Mutated	0.0835	0.0882	0.0813	0.0922
WhatsApp Mutated	0.0881	0.0711	0.0841	0.0696
Viber Mutated	0.0766	0.0442	0.0719	0.0742
YouTube Mutated	0.0559	0.0769	0.0683	0.0723

Table 19. MTU and Random Padding Results

Algorithm	SVM	BT	KNN	RF
Original % Accuracy	76.7	90	83.9	91.1
% Accuracy of MTU Quantization	31.8	22.3	25.4	20.3
% Accuracy of Random Padding	52.3	41.9	43.6	40.4

compromising anonymity and secrecy. Published studies to date have not come up with an ideal solution to prevent such attacks. In this article, we investigated optimal ways to protect privacy against traffic analysis.

First, we presented a practical method for obfuscation and morphing that reduced a classification accuracy from 91.1% to 0.22% with 11.86% overhead. This technique entails mutating packet lengths of a source app to those of a target app with similar bin probability. We shape the source app traffic to obfuscate its characteristics and make their distribution like that of another target app packet lengths. This complicates the ability of attackers to distinguish between the modified app traffic and the original one using statistical analysis.

Then, we formulated network obfuscation of the traffic analysis attack as a mathematical optimization problem. We proposed an algorithm to solve the problem of selecting the optimal target app as well as the optimal target packet length to mutate to. We designed a constructive solution that achieves maximum obfuscation and minimum padding overhead. This approach outperforms the other state-of-the-art defense solutions against traffic analysis, decreasing a classification accuracy of 91.1% to 1.76%, with only 0.73% overhead.

Our models are well intended to both secure traffic and preserve the computational requirements. They are appropriate for network devices with limited plans, especially since they reduce overhead in terms of data and bandwidth. Another advantage of these methods is that matrices mapping the mutation needed to modify the source app into the target app can be realized offline.

Accordingly, our schemes can run in a dynamic online manner and the entire process induces minimal latency.

In the future, we plan to integrate and implement our designs in a general network architecture.

REFERENCES

- [1] Comscore. 2018. Global digital future in focus 2018. *Comscore White Paper* 6, (2018).
- [2] Statista.com. 2019. The Statistics Portal for Market Data, Market Research and Market Studies. Retrieved from <https://www.statista.com/>. [Accessed 15 August 2019.]
- [3] Liftoff. 2019. Mobile App Trends Report. Retrieved from <https://liftoff.io/> [Accessed 30 Jan. 2020.]
- [4] Zhang Chaoyun, Paul Patras, and Hamed Haddadi. 2019. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys & Tutorials* 21, 3 (2019), 2224–2287.
- [5] Anna L. Buczak and Erhan Guven. 2015. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials* 18, 2 (2015), 1153–1176.
- [6] Erfan A. Shams and Ahmet Rizaner. 2018. A novel support vector machine based intrusion detection system for mobile ad hoc networks. *Wireless Networks* 24, 5 (2018), 1821–1829.
- [7] Mowei Wang, Cui Yong, Wang Xin, Xiao Shihan, and Jiang Junchen. 2017. Machine learning for networking: Workflow, advances and opportunities. *IEEE Network* 32, 2 (2017), 92–99.
- [8] Luke Scime and Jack Beuth. 2018. Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm. *Additive Manufacturing* 19 (2018), 114–126.
- [9] Zubair Md Fadlullah et al. 2017. State-of-the-art deep learning: Evolving machine intelligence toward tomorrow’s intelligent network traffic control systems. *IEEE Communications Surveys & Tutorials* 19, 4 (2017), 2432–2455.
- [10] He Ying et al. 2017. Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach. *IEEE Communications Magazine* 55, 12 (2017), 31–37.
- [11] Nguyen Cong Luong et al. 2019. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials* 21, 4 (2019), 3133–3174.
- [12] Georgi Ajaiya, Imad H. Elhadj, Ali Chehab, Ayman Kayssi, Marc Kneppers. 2018. Mobile apps identification based on network flows. *Knowledge and Information Systems* (2018) 1–26.
- [13] Louma Chaddad, Ali Chehab, Imad H. Elhadj, and Ayman Kayssi. 2018. App traffic mutation: Toward defending against mobile statistical traffic analysis. *IEEE INFOCOM on Computer Communications Workshops (INFOCOM WKSHPS)*.
- [14] Ahmet Aksoy, Sushil Louis, and Mehmet Gunes. 2017. Operating system fingerprinting via automated network traffic analysis. In *IEEE Congress on Evolutionary Computation (CEC’17)*. Donostia, San Sebastián, Spain, June 5–8, 2017. 2502–2509. <https://doi.org/10.1109/CEC.2017.7969609>.
- [15] Xu, Yixiao et al. 2018. A multi-tab website fingerprinting attack. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACM, 327–341.
- [16] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Goethem, and Wouter Joosen. 2018. Automated website fingerprinting through deep learning. In *25th Annual Network and Distributed System Security Symposium (NDSS’18)*. San Diego, CA, February 18–21, 2018. <https://arxiv.org/abs/1708.06376>
- [17] Kumar Ayush and Teng Joon Lim. 2019. Early detection of mirai-like IoT bots in large-scale networks through sub-sampled packet traffic analysis. In *Future of Information and Communication Conference*. Springer, Cham, 847–867.
- [18] Jishen Yu, Feng Liu, Wenli Zhou, and Hua Yu. 2014. Hadoop-based network traffic anomaly detection in backbone. In *IEEE 3rd International Conference on Cloud Computing and Intelligence Systems (CCIS’14)*, Shenzhen, China, November 27–29, 2014. 140–145. <https://doi.org/10.1109/CCIS.2014.7175718>
- [19] Conti Mauro, Luigi Vincenzo Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. 2015. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions on Information Forensics and Security* 11, 1 (2015), 114–125.
- [20] Conti Mauro, 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.
- [21] Bakhshandeh Atieh and Zahra Eskandari. 2018. An efficient user identification approach based on Netflow analysis. In *IEEE 15th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*. 1–5.
- [22] Shui Yu. 2016. Big privacy: Challenges and opportunities of privacy study in the age of big data. *IEEE Access* 4 (2016), 2751–2763.

- [23] Alessandro D'Alconzo, Idilio Drago, Andrea Morichetta, Marco Mellia, and Pedro Casas. 2019. A survey on big data for network traffic monitoring and analysis. *IEEE Transactions on Network and Service Management* 16, 3 (2019), 800–813.
- [24] Song Shi and Ruixuan Li. 2019. Traffic identification method based on multiple probabilistic neural network model. *Neural Computing and Applications* 31, 2 (2019), 473–487.
- [25] Sam Leroux, Steven Bohez, Pieter-Jan Maenhaut, Nathan Meheus, Pieter Simoens, and Bart Dhoedt. 2018. Fingerprinting encrypted network traffic types using machine learning. In *IEEE/IFIP Network Operations and Management Symposium (NOMS'18) Taipei, Taiwan, April 23-27*. 1–5.
- [26] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. 2016. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security Heraklion, Greece, September 26-30*. Springer, Cham, 27–46.
- [27] Mohsen Imani, Mohammad Saidur Rahman, and Matthew Wright. 2018. Adversarial traces for website fingerprinting defense. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security Toronto, Canada, October 15-19*. 2225–2227.
- [28] David Lu, Sanjit Bhat, Albert Kwon, and Srinivas Devadas. 2018. DynaFlow: An efficient website fingerprinting defense based on dynamically-adjusting flows. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society Toronto, Canada, October 15-19*. ACM, 109–113.
- [29] Abolfazl Diyanat, Ahmad Khonsari, and Seyed Pooya Shariatpanahi. 2016. A dummy-based 2014 approach for preserving source rate privacy. *IEEE Transactions on Information Forensics and Security* 11, 6 (2016), 1321–1332.
- [30] Parvathinathan Venkatasubramaniam, Ting He, and Lang Tong. 2008. Anonymous networking amidst eavesdroppers. *IEEE Transactions on Information Theory* 54, 6 (2008), 2770–2784.
- [31] Alfonso Iacovazzi and Andrea Baiocchi. 2012. Padding and fragmentation for masking packet length statistics. In *International Workshop on Traffic Monitoring and Analysis Vienna, Austria, Mar 12, 2012*. Springer, Berlin, 85–88.
- [32] Alfonso Iacovazzi and Andrea Baiocchi. 2013. Investigating the trade-off between overhead and delay for full packet traffic privacy. *IEEE International Conference on Communications Workshops (ICC'13) Budapest, Hungary, June 9-13*. 1345–1350.
- [33] Alfonso Iacovazzi and Andrea Baiocchi. 2012. From ideality to practicability in statistical packet features masking. In *8th International Wireless Communications and Mobile Computing Conference (IWCMC'12) Limassol, Cyprus, August 27-31*. IEEE, 456–462.
- [34] 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 *IEEE Symposium on Security and Privacy San Francisco, California, United States, May 20-23*. 332–346.
- [35] Charles V. Wright, Scott E. Coull, and Fabian Monrose. 2009. Traffic morphing: An efficient defense against statistical traffic analysis. In *Proceedings of the Network and Distributed Security Symposium (NDSS'09) San Francisco, California, United States, February 8-11*. IEEE, 1–14.
- [36] Alfonso Iacovazzi and Andrea Baiocchi. 2010. Optimum packet length masking. In *International Teletraffic Congress Amsterdam, The Netherlands, September 7-9*. 1–8.
- [37] Alfonso Iacovazzi and Andrea Baiocchi. 2014. Internet traffic privacy enhancement with masking: optimization and tradeoffs. *IEEE Transactions on Parallel Distributed Systems* 25, 2 (2014), 353–362.
- [38] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2019. Keeping the smart home private with smart (er) IoT traffic shaping. *Proceedings on Privacy Enhancing Technologies* 3 (2019), 128–148.
- [39] Louma Chaddad, Ali Chehab, Imad H. Elhadj, and Ayman Kayssi. 2019. AdaptiveMutate: A technique for privacy preservation. *Digital Communications and Networks* 5, 4 (2019), 245–255.
- [40] Louma Chaddad, Ali Chehab, Imad H. Elhadj, and Ayman Kayssi. 2019. Mobile traffic anonymization through probabilistic distribution. In *22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN'19) Paris, France, February 18-21*. IEEE, 242–248.
- [41] Saman Feghhi and Douglas J. Leith. 2016. A web traffic analysis attack using only timing information. In *IEEE Transactions on Information Forensics and Security* 11, 8 (2016), 1747–1759.
- [42] Ivan Homoliak, Daniel Ovsonka, Karel Koranda, and Petr Hanacek. 2014. Characteristics of buffer overflow attacks tunneled in http traffic. In *2014 International Carnahan Conference on Security Technology (ICCST'14)*. IEEE, 1–6.
- [43] Fabian Schneider, Anja Feldmann, Balachander Krishnamurthy, and Walter Willinger. 2009. Understanding online social network usage from a network perspective. In *Proceedings of the ACM Internet Measurement Conference Chicago, Illinois, United States, Nov 4*. 35–48.
- [44] Brendan Saltaformaggio et al. 2016. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic. In *10th USENIX Workshop on Offensive Technologies (WOOT'16) Austin, Texas, USA, August 8-9*. 1–10.

- [45] Reyhane Attarian, Lida Abdi, and Sattar Hashemi. 2019. AdaWFP: Adaptive online website fingerprinting attack for Tor anonymous network: A stream-wise paradigm. *Computer Communications* 148 (2019), 74–85
- [46] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *ACM Conference on Computer and Communications Security Toronto, Canada, October 15-19*. 1928–1943.
- [47] David Wagner, Bruce Schneier. 1996. Analysis of the SSL 3.0 protocol. *2nd USENIX Workshop on Electronic Commerce Proceedings* 1, 1 (1996), 29–40.
- [48] 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 Chicago, Illinois, USA, October 17*. 103–114.
- [49] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. 2019. Triplet fingerprinting: More practical and portable website fingerprinting with N-shot learning. *ACM Conference on Computer and Communications Security London United Kingdom, November 11-15*. 1131–1148.
- [50] Li Shuai, Huajun Guo, and Nicholas Hopper. 2018. Measuring information leakage in website fingerprinting attacks and defenses. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security Toronto, Canada, October 15-19*. 1977–1992.
- [51] 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 Scottsdale, Arizona, United States, November 3*. 131–134.

Received March 2020; revised November 2020; accepted December 2020