Thwarting Wi-Fi Side-Channel Analysis through Traffic Demultiplexing

Fan Zhang[1], Wenbo He[4], Yangyi Chen[2], Zhou Li[2], XiaoFeng Wang[2], Shuo Chen[3] and Xue Liu[4]

fzhang2@unl.edu, wenbohe@cs.mcgill.ca, {yangchen, lizho, xw7}@indiana.edu
shuochen@microsoft.com, xueliu@cs.mcgill.ca

Department of Electrical Engineering, University of Nebraska-Lincoln, NE, USA[1]
School of Informatics and Computing, Indiana University, IN, USA[2]
Microsoft Research, Microsoft Corporation, Redmond, WA, USA[3]
School of Computer Science, McGill University, Montreal, Quebec, Canada[4]

Abstract

Side-channel information leaks have been reported in various online applications, especially, in wireless local area networks (WLANs) due to the shared-medium nature of wireless links and the ease of eavesdropping. Even when Wi-Fi traffic is encrypted, its characteristics are identifiable, which can be used to infer sensitive user activities and data. Existing countermeasures do not offer effective and efficient protection: packet padding and traffic morphing often bring in substantial communication overheads; attempts to anonymize user identifiers are vulnerable to the analysis based upon traffic statistics, which allows the adversary to link traffic traces to individual users.

In this paper, we present a new technique, called *traffic demultiplexing*, which offers effective protection against Wi-Fi traffic analysis without incurring noticeable overhead and performance degradation. Our approach utilizes Media Access Control (MAC) layer virtualization and packet scheduling over multiple virtual MAC interfaces to shape the traffic on each virtual MAC interface, so as to hide the original traffic characteristics. Different from the higher-layer defensive approaches designed for specific applications, traffic demultiplexing operates at the MAC layer and therefore provides a general defense for various applications. In addition, it is transparent to users and other protocol stacks. We implemented our technique over Multiband Atheros Driver for Wi-Fi (MadWifi) and evaluated it in real WLAN environments. Our experimental study demonstrates that traffic demultiplexing is effective and efficient in defending against traffic analysis attacks and also easy to deploy.

1 Introduction

Side-channel information leaks are pervasive in different communication scenarios, including web browsing [1, 2], video-streaming [3], voice over-IP (VoIP) applications [4] [5], and secure shell (SSH) [6]. These information leaks are mostly caused by analyzing statistical characteristics of encrypted traffic, such as distributions of packet sizes, inter-packet timings and others. Adversaries are found to be able to tailor traffic analysis techniques to seriously threaten user privacy, even when the traffic is protected by up-to-date encryption techniques. Just through the analysis on traffic characteristics, for example, adversaries can identify online activities (e.g., web-browsing, chatting, online gaming, online video, and downloading) [7–9], then infer the sources of web pages or contents of those online activities [1,10], and further obtain sensitive information, such as health records, family incomes, and investment strategies etc [2].

The threat of traffic analysis is particularly serious in wireless networks, which has been widely deployed in residential, hotspot, and campus environments for Internet access. Due to the shared-medium nature of wireless links, adversaries can easily eavesdrop on a specific user's traffic using sniffer software (e.g., Wireshark, Aircrack-ng). For example, it was reported that the Google street view team "inadvertently" collected Wi-Fi traffic snippets that people sent over wireless links [11]. In another example, prior research shows that through sniffing Wi-Fi traffic, a stranger on the street can glean enterprise employees' search queries, despite the protection of Wi-Fi encryptions (WPA/WPA2) [2].

To defend against traffic analysis, techniques have been developed to make traffic characteristics less identifiable. Commonly used strategies alter the distribution of packet size through padding packets [1, 2], faking superfluous packets, and chopping packets into fixed size segments [12]. One can also use traffic morphing techniques [13] to make the statistical features of one class of traffic look like those of another class. These approaches, however, often introduce significant communication overheads [2], and requires remote servers (e.g., web servers and multimedia servers) to change the packet size distribution for each application or each user, which make their deployment questionable.

An alternative approach is to use anonymous communication [14, 15], the identifier-free approach [16] and pseudonym [17, 18] to prevent adversaries from associating individual packets with a specific user. However, prior research shows traffic characteristics (e.g., packet distribution and received signal strength indicator (RSSI)) may still allow an eavesdropper to identify the party in communication [19], thereby disclosing his/her sensitive information.

To achieve both effective and efficient defense against traffic analysis, we present in this paper a novel design, called *traffic demultiplexing*. Our idea is to partition a Wi-Fi traffic flow into *snippets*, each composed of a subset of the packets in the flow, when transmitting these packets through different virtual Media-Access-Control (MAC) interfaces to a Wi-Fi channel. These snippets are later reassembled at a Wi-Fi Access Point (AP), such as a wireless router, before the communication is relayed to the Internet. This presents to a Wi-Fi eavesdropper multiple traffic flows from different sources, each with traffic characteristics sufficiently deviated from the original flow and apparently independent of those from other MAC interfaces. These flows can be further adjusted to look like being produced by different Internet activities. As a result, the adversary can be cheated into believing that multiple users are running different programs on different systems. Given this confusion of application contexts, private user information becomes much more difficult to extract. Compared with prior approaches, our techniques avoid aggressive padding to achieve a similar level of protection, thereby performing more efficiently. To make this happen, we developed a new MAC layer traffic partition approach that includes MAC layer virtualization and dynamic packet scheduling over multiple virtual MAC interfaces, and conducted a thorough evaluation of the techniques. We provide a demo for our proposed technique (http://www.youtube.com/user/fccherry091?feature=mhum).

We summarize the contributions of the paper below:

• *Novel defense against Wi-Fi side-channel analysis.* Compared with existing techniques, our novel design achieves a much more cost-effective defense against traffic analysis over wireless links. Specifically, traffic demultiplexing operates at the MAC layer, hence provides a more generic protection for the applications than the high-level mitigation designed specifically for these applications [2, 13]. Further, our design partitions the traffic over wireless links into snippets and sends the snippets through multiple virtual MAC interfaces to hide their traffic characteristics, which avoids the significant overhead other approaches incur for packet padding and noise adding. Finally, the new technique grants Wi-Fi users a fine-grained control on how their traffic is divided, and offers new scheduling strategies to assign traffic snippets to different interfaces. This makes a Wi-Fi eavesdropper more difficult at identifying the features of the original traffic.

•*Implementation.* We implemented a prototype of our techniques on the Multiband Atheros

Driver for Wi-Fi (MadWifi). Through MAC layer virtualization, virtual wireless interfaces can be dynamically created and their attributes can be configured (e.g., MAC address, transmission power and data rate). Our approach treats each virtual interface as a regular network interface, and schedules network traffic onto these interfaces according to a set of user-defined packet scheduling algorithms, including ours that are designed to optimize traffic demultiplexing. The prototype we implemented includes a per-packet-based transmission power control (TPC) technique to prevent adversaries from inferring user identification through the strengths of Wi-Fi signals, and the countermeasures to thwart attempts to link virtual interfaces to a given user. Also, our technique is built to be transparent to upper layers and be compatible with the IEEE 802.11 standards.

•*Evaluations.* We evaluated traffic demultiplexing in real WLAN environments under different traffic analysis attacks. Experiments show that the technique effectively suppresses traffic analysis: it reduced the classification accuracy from 83.24% to 44.21% for online activities, from 30.6% to 5.0% for the web page identification attack, and from 78.3% to 60.0% for language identification in VoIP traffic, without noticeable overhead. Meanwhile, our study demonstrates that traffic demultiplexing does not incur noticeable network performance degradation and scales well when deployed in real WLANs.

The preliminary idea of scheduling traffic onto multiple interfaces has been discussed in [20], where a naive scheduling algorithm was presented, and packets are scheduled only according to the packet size range. However, with the proposed scheduling algorithm in [20] an attacker may easily infer which virtual interfaces are used by a specific user through combination analysis. On the other hand, several implementation issues are also ignored in [20], such as which communication layers should be changed to implement the demultiplexing idea, what protocols should be affected and adapted with traffic demultiplexing, how to prevent against timing analysis attack by an attacker to figure out which interfaces belong to which user, how network performance will be affected if users adopt multiple interfaces in reality, and how to make the proposed idea against a wide range of side-channel information leak in WiFi networks besides the online activity information leak, including web page identification, and the leak of VoIP features.

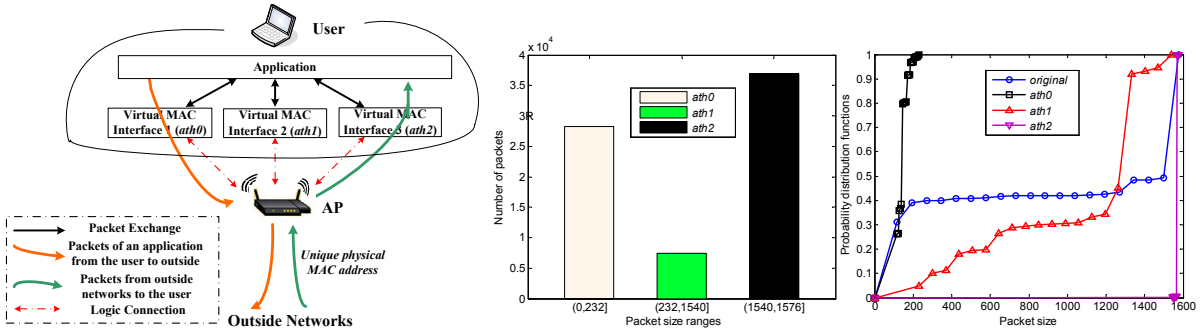
The remainder of this paper is organized as follows. We present the background of our work in Section 2. We then describe the detailed design and implementation of traffic demultiplexing in Section 3. Section 4 presents the security analysis. In Section 5, we evaluate traffic demultiplexing through experimental testing in real WLANs. We summarize the related work in Section 6. Finally, we conclude the paper in Section 7.

2 Background

2.1 Attack Model

Traffic analysis techniques have been studied for decades [1–10]. The adversary assumption is that the attacker can eavesdrop on communication traffic, and observe its characteristics, such as packet size, interarrival timing and Wi-Fi signal strengh, etc. Using the traces of these characteristics, the adversary can classify them to infer sensitive user information through various machine learning techniques, such as Support Vector Machine (SVM), Neural Network (NN), Bayesian techniques and Hidden Markov Models (HMM). Prior research shows that these techniques can succeed on a wide range of online applications [5, 9, 21, 22]. In our research, we consider the following three common attack scenarios, which are extensively studied in prior work, to evaluate the efficacy of our new technique:

- *Identification of user online activities.* Traffic characteristics, such as packet interarrival time and packet size, can be used to profile users' actual online activities (e.g., web-



(a) Data transmission in traffic demultiplexing (b) Traffic on each interface (c) PDF on each interface

Figure 1: An example of traffic demultiplexing. (a) Packets from a single application are dynamically switched over multiple MAC interfaces. For example, an outgoing packet encapsulated by the MAC address of *ath0* is sent to an access point (AP). The response from the outside networks is sent to the user through another virtual MAC interface *ath2*. (b) Original traffic is distributed into three virtual interfaces. *ath0* transmits packets with sizes smaller than 232 bytes, packets with size larger than 1540 bytes are dispatched by *ath2*, and the rest are sent through *ath1*. The packet sizes on three virtual interfaces are orthogonal. (c) We see that three virtual interfaces have very different packet distributions (e.g., average packet size of the original traffic is 962.04 bytes, that of three virtual interfaces are 143.88, 1061.90 and 1568.00 bytes, respectively).

browsing, chatting, online game, online video, and downloading), even when the traffic is encrypted [8, 9].

- *Identification of web pages.* Sun, et al. [1] and Liberatore, et al. [10] show that it is possible to accurately identify a web page by using only the sizes of the packets and the direction which they passed through (i.e., from or to the client) in the encrypted traffic.
- *Identification of VoIP features.* Wright, et al. [4] show that when VoIP packets are compressed with variable bit rate (VBR) encoding schemes and encrypted with a length-preserving stream cipher, it is possible to determine the language spoken in the encrypted conversation by running classifiers (e.g., HMM and Bayesian) on "bit rate."

Note that different from those prior studies, which mainly focus on IP packets, we look at the threat on the MAC layer: our approach works on frame sizes without knowing their end-to-end information.

2.2 Objectives

Existing countermeasures against Wi-Fi side-channel leaks either incur significant overheads and cause performance degradation or are found to be less effective. For example, *physical space security and jamming* [23, 24] aim to reduce the number of packets that can be overheard by eavesdroppers but introduce interference with legitimate communication [19]. As another example, *anonymity channels* [14, 15] and *the identifier-free approach* [16] were used to conceal user identifiers from a Wi-Fi eavesdropper. However, prior research shows that these approaches are vulnerable to physical layer fingerprinting [19] (e.g., RSSI values), which allows the adversary to associate Wi-Fi packets with a specific user. In addition, the overheads of encryption and key management of these techniques were found to be substantial, resulting in a throughput degradation of 10% according to a prior study [16]. Other conventional defense such as *traffic padding*, *packet padding*, and *traffic morphing* could also incur a large overhead, as reported by prior research [1, 2, 12, 13].

A practical defense against Wi-Fi traffic analysis is expected to have the following properties:

- Effectiveness: the defense should significantly reduce the accuracy of a traffic analysis.
- Efficiency: the technique is expected to be efficient, without incurring noticeable overheads and performance degradation.
- Generic: it is highly desired that the new approach offers a more generic protection to a wide range of applications against various side-channel analyses.
- Compatibility: the technique should be compatible with the existing standards and protocols, and transparent to users and the application developers.

To achieve these objectives, we developed a new technique based upon *traffic partition* [25, 26], a technique originally designed for privacy preserving data collection and publication. Prior research such as *frequency hopping* [27] partitions wireless traffic into multiple flows and transmits them through different frequencies. In contrast, our traffic demultiplexing approach creates several parallel virtual wireless interfaces, which can be associated with different applications, and also allows per-packet-based scheduling to assign each packet to a wireless interface. This treatment alters the inter-packet timings, packet size distributions and other traffic features, and therefore offers a more effective protection against traffic analysis. In addition, our new approach does not require aggressively padding packets or adding cover traffic, and therefore avoids substantial communication overhead.

Figure 1 presents an example of traffic demultiplexing, which partitions and transmits the Wi-Fi traffic produced by a BitTorrent (BT) application through three virtual interfaces (as shown in Figure 1(a)): *ath0* sends packets smaller than 232 bytes; *ath2* delivers those larger than 1540 bytes, and the rest goes through *ath1*. Figure 1(c) shows the *probability distribution function* (PDF) of packet sizes on each interface, which demonstrates how our approach changes traffic characteristics in terms of packet size distributions over individual interfaces. As we can see from the figure, these traffic characteristics become dramatically distinctive from those of the original traffic.

3 Traffic Demultiplexing

3.1 Overview

In this section, we elaborate the design and implementation of our demultiplexing techniques.

Figure 2 sketches the modules and network protocol stack of our approach. *Service module* controls the communication between a client and a Wi-Fi Access Point (AP) for configuring virtual MAC interfaces. *MAC-layer virtualization* supports the creation and management of multiple virtual MAC interfaces over a single physical interface. Each interface is configured with different MAC addresses and treated as a fully functional, regular network interface. This virtualization can be turned on and off according to the user's privacy policies.

Bridge layer is a veneer placed above the virtualized MAC layer to provide transparent networking interfaces for its upper layers. It is responsible for dispatching outbound packets to multiple virtual interfaces through the *demultiplexing scheduler* and concatenating inbound packets from these interfaces through *MAC translation*. Demultiplexing scheduler, which ensures that schedules packets through different interfaces, is discussed in Section 3.4.

We implemented our design into a prototype by modifying the latest MadWifi (madwifi-ng), a popular WLAN driver [28], both on the AP and on the client side. Our implementation works well with Wired Equivalent Privacy (WEP), WPA/WPA2, and dynamic host configuration protocol (DHCP), etc. It is compatible with the IEEE 802.11 protocol set: traffic demultiplexing on a client or an AP does not interfere with the communication of the parties that do not use it.

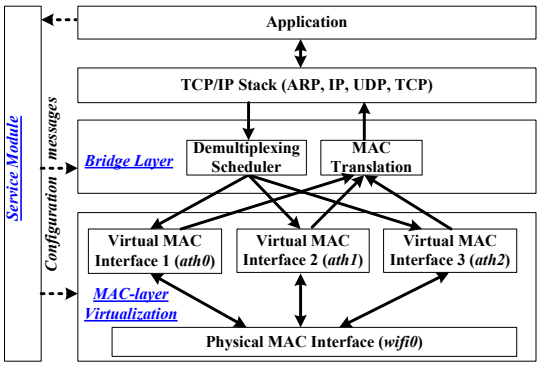


Figure 2: Network stack with three virtual interfaces. Our modifications include service module, bridge layer, and MAC-layer virtualization.

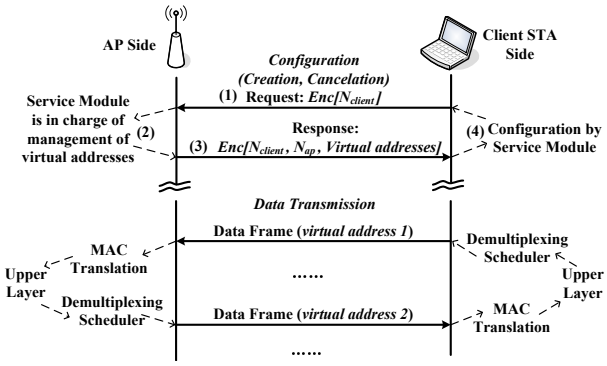


Figure 3: Communication process over multiple virtual wireless interfaces.

3.2 Creating Virtual MAC Interfaces

As illustrated in Figure 3, virtual MAC interfaces are created by a two-way handshake through an encrypted channel, which includes the following four steps. (1) A client sends out a request to create virtual interfaces with a nonce N_{client}, encrypted with a symmetric key shared by the client and the AP. (2) Upon receiving the request, the AP first chooses the number of virtual interfaces to create, denoted as I, which is determined by the privacy policies set by the user and the available resource. The privacy policy can be as simple as whether a specific application, indicated by its Transfer Control Protocol (TCP) port numbers, needs this demultiplexing service. Each virtual interface created is given an unused MAC address randomly selected by the AP. More specifically, a MAC address has 48 bits, with the first 24 bits used to identify the vendor. Our approach ensures that only the 24-bit sequences related to real vendors are picked up when assigning virtual addresses, thereby making the virtual interfaces look realistic. To keep track of these addresses, the AP maintains a look-up table to map virtual addresses to the physical MAC addresse of the user's adapter. (3) Then, the AP sends the client an encrypted reply including both the nonce and the assigned virtual MAC addresses. (4) Once the client receives the response, it checks whether the nonce matches the one it includes in its request. If so, it begins to virtualize MAC interfaces using received MAC addresses. This virtualization was implementable through modifying the Wi-Fi drivers such as MadWifi. Note that a client also needs to maintain a table to map its virtual addresses to physical ones.

To cancel a virtual interface, the client goes through a similar handshaking process to request the AP to release the interface. In this way, the AP is able to actively recycle the resources and dynamically configure virtual interfaces. This two-way handshake can be piggybacked in the

management frames (e.g., probe frames and association frames) specified by the IEEE 802.11. However, these frames are not encrypted by current APs and Wi-Fi clients. As a result, we had to implement the protocol into the encrypted data frames of IEEE 802.11, which also require the client to be first authenticated by the AP.

3.3 Communication with Virtual Interfaces

Traffic demultiplexing supports virtual MAC interfaces on a wireless card, with different MAC addresses working in a station mode. It dynamically specifies device attributes for each packet and makes each virtual interface look like a different wireless card. Also, we modified the sending and receiving procedures of the MadWifi on both the AP and client sides. The details of this implementation are described as follows.

Sending procedure. For sending frames, we set the device attributes (e.g., MAC address, priority transmission queue, transmission power, and data rate) of each virtual interface dynamically for each packet passing through. First, the demultiplexing scheduler assigns a virtual MAC interface to a packet. As shown in Figure 4, a packet is dispatched to the virtual interface *ath0*. Then *ath0* encapsulates the packet to a MAC frame. Note that the current network adapter ensures that the size of an IP packet is within the maximum transmission unit (MTU) specified at the MAC layer, and therefore re-fragment of the packet becomes unnecessary. Such an encapsulation was implemented in the function "*ieee*80211_*encap*". After that, the virtual interface adjusts the transmission power and data rate before sending the packet, as a real wireless card does.

Receiving procedure. According to the IEEE 802.11, a client drops the packets whose destination MAC addresses do not match the client's MAC address. In our implementation, we set the receiver filter "*rfilt*" of the physical wireless interface to promiscuous mode ($HAL\_RX\_FILTER\_PROM$) and then utilized a software filter to only accept the packet when its destination address is one of the virtual MAC addresses the client possesses. As illustrated in Figure 4, the client performs *MAC translation* by replacing the virtual MAC address in the *destination address field* with the unique physical MAC address before the packet is delivered to the upper-layer. This translation makes our changes to the MAC layer transparent to the protocols working on the upper layers, especially to Address Resolution Protocol (ARP) that maps an IP address to a physical MAC address. We implemented MAC translation in the decapsulation function "*ieee*80211_*decap*."

An AP keeps record of the MAC addresses of the clients connecting to it (or "associated" with it). For each packet received, the AP checks its source MAC address and drops the packet if the address is not known. In our research, we modified "*ieee*80211_$find_rxnode$", a function that searches for such associated addresses, to replace virtual MAC addresses with their corresponding physical counterparts. We also enabled the AP to perform the MAC translation as described before.

In summary, traffic demultiplexing is transparent to upper-layer and other protocol stacks. Also, our modification on the MAC layer does not affect user experience and is compatible with the IEEE 802.11 standard.

ACK Loss vs. Rate Control. In Wi-Fi settings, data rate is adaptive to the channel quality measured by packet loss rate, which is estimated based on the MAC-layer acknowledgement (ACK) frames. These ACK frames are sent by the wireless card automatically when the hardware receives the packets labeled its physical MAC address. However, the packets sent to virtual MAC addresses cannot trigger the ACK frames. Without MAC-layer ACK frames in standard IEEE 802.11 protocol, a sender (i.e., AP) has to retransmit the packets several times until reaching maximum retry times, which would make the sender assume that the wireless link was bad and the packet loss rate was quite high. As a result, the sender would set the data rate as low as "1Mbps". To avoid such a network performance degradation, we can use

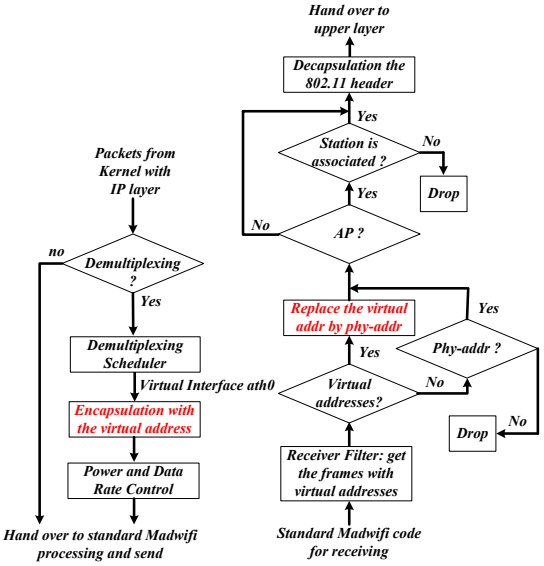


Figure 4: Packets flow on receiver and sender sides

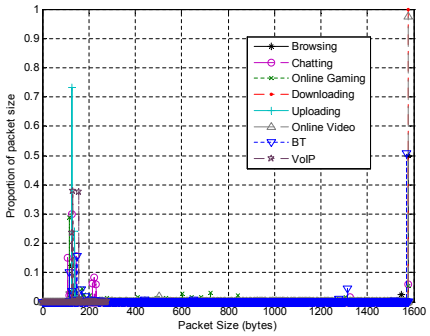


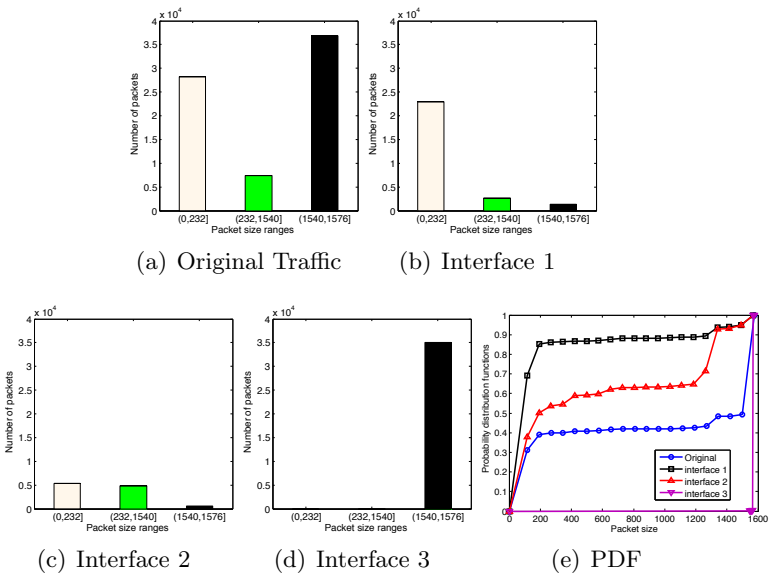
Figure 5: Packet size proportion of various applications (from an AP to a client)

fake ACK frames generated by the network driver, or disable the ACK frames, which in turn disables the rate adaptation. For simplicity, our prototype simply disables the automatic MAC-layer ACK by changing the Atheros configuration register number $0x8048$ and setting the flag HAL_TXDESC_NOACK. Further, we set the attribute, tx_rate, in a bunch of packets to dynamically control the data rate.

3.4 Demultiplexing Scheduler

The demultiplexing scheduler aims to hide the characteristics of the original traffic by dispatching packets through different virtual interfaces. This objective, however, cannot be well served by naive scheduling policies such as ***Random Algorithm (RA)*** (randomly scheduling a packet to a virtual interface) and ***Round-Robin (RR)*** (scheduling packets in the order of virtual interfaces), as the traffic characteristics (e.g. packet size distribution) can still be recovered by monitoring a wireless interface sufficiently long: after all, the sniffed traces represent an unbiased sample from the original traffic. Hence, more effective scheduling techniques need to be developed to transform the traffic features beyond recognition. Here we elaborate the approaches designed and implemented in our research.

The traffic characteristics most extensively used in traffic analysis include packet size and



(a) Original Traffic (b) Interface 1

(c) Interface 2 (d) Interface 3 (e) PDF

Figure 6: Traffic demultiplexing with MD for a BT application. (a) The original packet size distribution is $[0.39, 0.1, 0.51]$. (b) Interface 1 simulates chatting with the distribution $[0.8515, 0.0985, 0.05]$. (c) Interface 2 resembles online gaming with the distribution $[0.5005, 0.4495, 0.05]$. (d) Interface 3 is disguised as downloading with the distribution $[0, 0, 1]$. We see that the actual distributions of virtual interfaces is very close to the target probability distribution. (e) The correlation of each interface is not obvious, and packet distribution on three interfaces are distinct from the original traffic.

packet interarrival time. Our approach schedules the packets within the same traffic flow through different interfaces, which automatically causes the interarrival times observed from individual virtual interfaces to deviate from that measured from the physical interface. Therefore, the focus of our research is on changing the packet size distributions associated with individual virtual interfaces to deceive the eavesdropping adversary. Let I be the number of virtual interfaces and ℓ_{max} be the maximum packet size of incoming packets. Assume that there are L possible packet size ranges, $\{(0, \ell_1], (\ell_1, \ell_2], \cdots, (\ell_{L-1}, \ell_L]\}$, where $\ell_L = \ell_{max}$. Let p_j^i be the probability of packets whose size is among $(\ell_{j-1}, \ell_j]$ on the virtual interface i. We define the *target probability distribution* ϕ^i on interface i, where $\phi^i = [\phi_1^i, \phi_2^i, \cdots, \phi_j^i, \cdots, \phi_L^i]$. ϕ_j^i is denoted as the target probability of the packet size within $(\ell_{j-1}, \ell_j]$ on the virtual interface i. Preferably, we hope to make p_j^i as close to ϕ_j^i as possible. Therefore, given ϕ^i, how to design a demultiplexing scheduler to emulate the target is essentially an optimization problem as follows.

$$\begin{aligned} &\min \sum_{i=1}^{I} \sqrt{\left(\sum_{j=1}^{L} \left| \phi_j^i - p_j^i \right|^2 \right)} \\ &\text{sub} \sum_{i=1}^{I} p_j^i \mathcal{N}(i) = P_j \mathbb{N}; \sum_{i=1}^{I} \mathcal{N}(i) = \mathbb{N}; \\ &\sum_{j=1}^{L} \phi_j^i = 1; \sum_{j=1}^{L} p_j^i = 1; \\ &\phi_j^i, p_j^i \in [0, 1]; i \in [1, I]; j \in [1, L]; \end{aligned} \tag{1}$$

where $\mathcal{N}(i)$ is the number of packets on the interface i, $\mathbb{N}$ is the total number of packets, and we define P_j as the probability of packet size between $(\ell_{j-1}, \ell_j]$ in the original traffic. Note that different ϕ^i reflects different scheduling policy, and here we describe how to determine the

target packet size distribution, ϕ^i, to mislead adversaries and prevent them from linking virtual interfaces to a specific user.

Masquerading Demultiplexing (MD). This scheduling strategy is designed to make the traffic scheduled on a virtual interface look as if it is produced by a real application, so as to cheat the adversary into believing that the user is doing something else.

To determine ϕ^i for MD, let's first define the packet size ranges. Figure 5 shows the packet size distribution of various applications. We observe that the sizes of more than 95% packets fall into two ranges: [108, 232] and [1546, 1576]. Such packet size ranges are likely to be adopted by attackers for traffic analysis. Hence, we use the same packet size ranges to conceal the original packet size distribution, by taking $L = 3$ and the packet size ranges as $(0, 232]$, $(232, 1540]$, and $(1540, 1576]$. Through the measurements from traffic traces, we find that chatting contains mainly short packets and its packet size distribution over the above ranges is $[0.85, 0.1, 0.05]$, browsing has a large variance on packet size and its packet size distribution is $[0.3, 0.2, 0.5$, and downloading consists of large packets and its packet size distribution is $[0, 0, 1]$. Based on the observation on packet size distribution, the target distribution ϕ^i can be easily determined to masquerade interface i to a predefined application. With such ϕ^i, the MD is achieved by optimizing Equation 1. To achieve the best performance, the selection of target distribution ϕ^i is usually dependent on the original traffic.

Since traffic demultiplexing is running online, a scheduler needs to send a packet to a virtual interface without knowing the future traffic. The true optimality is hard to attain. As so, we introduce a heuristic algorithm to achieve a close optimal solution. This heuristic algorithm selects the interface which has the largest difference between ϕ^i_j and p^i_j to schedule the incoming packets.

Heuristic Algorithm for Traffic Demultiplexing:
Choose-VirtualInterface(L, I, ϕ^i_j)

1 **for** a coming packet s_k with its length $\mathscr{L}(s_k)$
2 **do** $j \leftarrow j$ satisfies $\mathscr{L}(s_k) \in (\ell_{j-1}, \ell_j]$
3 **for** $i \leftarrow 1$ **to** I
4 $\triangleright$ Compute p^i_j of interface i.
5 **do** $i' = \arg\max_{i \in [1,I]} (\phi^i_j - p^i_j)$
6 **return** i'

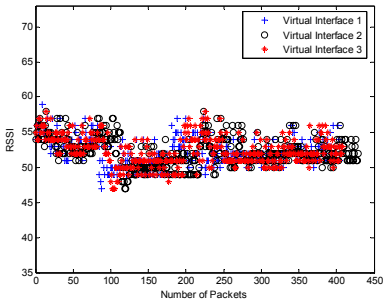
We next show an example in Figure 6 to illustrate the idea in MD, where we masquerade a BT flow as chatting, online gaming, and downloading onto three virtual interfaces, respectively. Therefore, we set the target packet size distribution as $\phi^1 = [0.85, 0.1, 0.05]$, $\phi^2 = [0.5, 0.45, 0.05]$, and $\phi^3 = [0, 0, 1]$. Through MD, the traffic on three virtual interfaces has distinct traffic characteristics and differs significantly from the original traffic.

In our design, APs and clients are allowed to choose their demultiplexing scheduling policies independently and change scheduling policies dynamically. Hence, users have the flexibility of using appropriate demultiplexing schedulers for their own privacy needs.

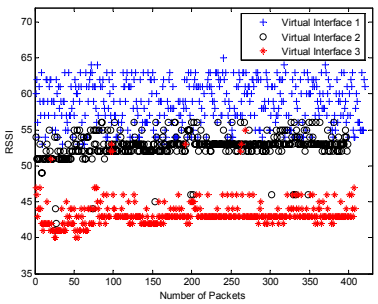
4 Security Analysis

In this section, we analyze the effectiveness of the proposed technique in defending against the attempts to link the virtual interfaces to specific users and restore the features of the original Wi-Fi traffic.

As described in Section 3, we have taken three measures to prevent an adversary from inferring virtual interfaces used by a specific user. (1) *Encrypted communication:* As shown in



(a) Without TPC



(b) With TPC

Figure 7: RSSI values of three virtual interfaces over one wireless card. (a) RSSI values vary in a narrow range and follow the same patterns without TPC. (b) Different virtual interfaces show different RSSI patterns with TPC.

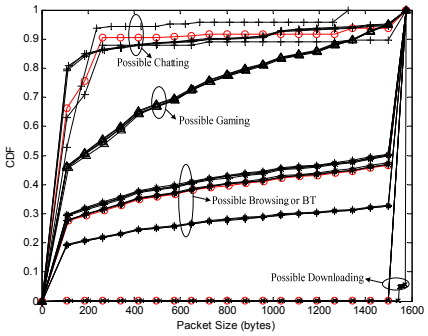


Figure 8: The CDF of possible applications by composition attacks. The real applications are in red line with the circle marker.

Section 3.2, we encrypt configuration messages of traffic demultiplexing, which contain information about the mapping between virtual and physical interfaces. (2) *Dynamic configuration of virtual interfaces:* We enable a dynamic configuration of the parameters for virtual interfaces (e.g., number of interfaces used, MAC addresses, power and rate levels). This makes it more difficult to correctly guess how many users are in the Wi-Fi network and which virtual interfaces are used by which user. (3) *Intelligent scheduling policies:* We dynamically update the scheduling algorithms for traffic demultiplexing in order to change the features of the traffic on different virtual interfaces continuously. Therefore, we prevent the adversaries from accumulating valid statistics to link the virtual interfaces to a specific user. The update can be triggered by the change of security requirement or the number of users in the Wi-Fi network.

Next, we analyze the protection our techniques offer in defending against power analysis, timing analysis and composition attack.

4.1 Against Power Analysis

Adversaries may attempt to use wireless signal strengths measured in multiple locations to infer a user's location and, therefore, associate packets to a specific user. Many existing defenses against traffic analysis are vulnerable to such a power analysis [19, 29–31]. To mitigate this threat, we adopt per-packet power control to obfuscate the observable features of received signal strength indicator (RSSI). Today's MadWifi drivers offer the support for per-packet-based transmission power control (TPC) with a granularity of $0.5dBm$ and low switching latency (less than 1 millisecond) [32, 33]. By adjusting TPC at a fine granularity when performing a per-packet scheduling over virtual interfaces, we were able to diversify the RSSI values on individual virtual interfaces, which prevents the adversary from associating these interfaces with a specific user.

Our implementation enables TPC in MadWifi by setting "$ath\_hal\_settpc(ah, 1)$." We adopted a TPC policy that gives a fixed transmission power to each virtual interface. For an outgoing packet, traffic demultiplexing first acquires the virtual interface it belongs to through the scheduler, and then configures the transmission power attribute, "$ni\_txpower$," of this packet according to the parameter set in the virtual interface. Also, we need to disable several adaptive mechanisms, such as the antenna diversity, rate selection algorithms and virtual carrier sensing (RTS/CTS) mechanism [32].

We tested the RSSI values of three virtual interfaces over one wireless card (Atheros 5212 chipset), as illustrated in Figure 7(a). The figure shows that without TPC, RSSI values vary in a narrow range and follow the same pattern on all three virtual interfaces. Hence, it is likely for

an adversary to associate these interfaces to a specific user. Using the TPC policy, Figure 7(b) demonstrates that the RSSI values on these interfaces exhibit different patterns. It appears that the traffic on these interfaces come from different users.

4.2 Resistance to Timing Attacks

The packet interarrival time is often used in traffic analysis to infer sensitive user data. Such a timing side-channel leak can be effectively suppressed by traffic demultiplexing. Specifically, by distributing packets to different virtual interfaces, our approach shuffles the packets from the original Wi-Fi flow over these interfaces, making the inter-packet timings observed from individual interfaces significantly different from those of the original flow. Here, we use an example to show how it works. In the example, we scheduled packets from a web browsing application onto three virtual wireless interfaces. The average packet interarrival time of the web browsing is $0.028s$. After traffic demultiplexing under MD (Section 3.4), the interarrival times on the interface 1, 2, and 3 became $0.330s$, $0.134s$, and $0.092s$ respectively. The average observed interarrival time on the interface 1 falls into the range of online chatting, which is typically between $0.3s$ to $5s$; the interarrival time on interface 2 is in the range of online gaming applications, mainly from $0.1s$ to $0.5s$; the interface 3 appears as if it is downloading data with a bandwidth of $17KBps$. Hence, the proposed traffic demultiplexing method is robust against the timing attacks. It implies that the packet interarrival time of Wi-Fi traffic can hardly be a reliable source of information for an eavesdropper, as it varies greatly depending on application settings, network conditions, service providers, and packet loss rates during sniffing.

4.3 Against Traffic Composition Analysis

An adversary may sniff the traffic on all the wireless interfaces and attempt to enumerate the combinations of the traffic flows. Demultiplexing using the MD algorithm (Section 3.4) makes the traffic on each virtual interface look like being produced by a real online application, thereby making it more difficult for the adversary to figure out whether our technique has been used in the wireless network, and to further link multiple virtual interfaces to a specific user.

Ideally, the Wi-Fi traffic produced by one application is divided into I sub-flows, each looking like that of a real (preferably different) application. Unfortunately, this cannot always be achieved in practice. What we can do here is adding a few noise packets to the flow so that it can exactly cover I flows of real applications. Specifically, consider the MD algorithm described in Section 3.4, we can make each sub-flow as close as possible to a real application through the elaborate selection of target distribution. Although it may be hard to satisfy all the cases, we can ensure that all but one of the I sub-flows will exhibit the traffic features of other real applications. Our approach further adds noise packets to the reminder sub-flow to morph it into another program's traffic. Note that the overheads incurred by this strategy are lower than those of the morphing technique proposed in prior research [13], simply because we only need to work on a sub-flow of the traffic instead of the whole traffic itself. This was verified in our experimental study (Section 5).

Fundamentally, both our approach and traffic morphing are kind of "camouflage" techniques that are designed to hide traffic features from the eavesdropper who are not aware of the use of the techniques. On the other hand, our research shows that even when this is not true, our approach can still hold its ground. Specifically, consider the adversary who assumes the presence of our demultiplexing technique in a WLAN and attempts to reconstruct the original traffic from observed flows. We studied this problem in a real-world scenario that involved three users doing online chatting, browsing and downloading respectively. In the experiment, the chatting and downloading traffic was transmitted through two virtual interfaces, and that of the browsing went through other three interfaces. The adversary who observed all 7 virtual

Table 1: Target distribution of MD for applications

App.	Target Distribution of MD		
	ϕ^1	ϕ^2	ϕ^3
br.	[0.85,0.1,0.05]	[0.5,0.45,0.05]	[0,0,1]
ch.	[0.5,0.45,0.05]	[0.3,0.2,0.5]	[0,0,1]
ga.	[0.85,0.1,0.05]	[0.3,0.2,0.5]	[0,0,1]
do.	[0.85,0.1,0.05]	[0.3,0.2,0.5]	[0,0,1]
up.	[0.85,0.1,0.05]	[0.3,0.2,0.5]	[0,0,1]
vo.	[0.85,0.1,0.05]	[0.3,0.2,0.5]	[0,0,1]
bt.	[0.85,0.1,0.05]	[0.5,0.45,0.05]	[0,0,1]

interfaces faced totally 127 ($\sum_{i=1}^{7} C_7^i$) possible combinations of the virtual interfaces. Among them, we found that at least 28 different combinations looked exactly like real traffic, including 6 chatting, 5 gaming, 6 downloading, and 11 browsing or BT communication. The cumulative distribution function (CDF) of these traffic compositions are presented in Figure 8. Given this anonymity set, it became very difficult for the adversary to figure out how many users were in the Wi-Fi network and what online activities were really happening there.

5 Evaluations

5.1 Real-world Deployment and Experiment Setup

In order to evaluate the effectiveness and efficiency of traffic demultiplexing, we conducted experiments in a real WLAN environment. A laptop with the prototype of traffic demultiplexing installed acts as an AP, which is equipped with a Proxim AP-2000 11b/g Cardbus Series (Atheros 5212 chipset) wireless card. Clients are equipped with Atheros wireless cards with different chipsets (AR5212 and AR928X), both allowing us to modify the MadWifi driver. The AP and clients run Ubuntu 10.04 with the kernel version of 2.6.32, and they all work in the 802.11g band. We vary the data rate in the experiments from 1Mbps to 54Mbps depending on network conditions. The AP supports the data encryption with WEP/WPA/WPA2 by running hostapd (hostapd-0.7.3) [34]. The sniffer program installed on another laptop collects the real traffic traces through the Intel Wireless Wi-Fi Link 4965AGN network card with the *Libpcap* [35] library.

Generally, we set the number of virtual interfaces $I = 3$. Seven popular applications are investigated. They are browsing (br.), chatting (ch.), gaming (ga.), downloading(do.), uploading(up.), online video (vo.), and Bit-Torrent (bt.). Packet sizes are divided into three ranges (i.e., $L = 3$): $(0, 232]$, $(232, 1540]$, and (1540, 1576]. For the given ranges, the target packet size distribution function ϕ^i in MD is listed in Table 1. In order to disguise, ϕ^1 is the distribution of *chatting* for every non-chatting application, and the ϕ^1 for chatting is the distribution of *gaming*. Similarly, ϕ^2 is $[0.3, 0.2, 0.5]$ to make the traffic look like *browsing* except for the original *browsing* and BT applications, which are simulated as *gaming*. ϕ^3 is set as $[0, 0, 1]$ to resemble a *downloading* application.

We measured the network performance of traffic demultiplexing, in terms of throughput and packet delay. We evaluated the effectiveness and efficiency of traffic demultiplexing with MD under three traffic analysis attacks. In comparison with other defensive approaches, such as packet padding and traffic morphing [13], we evaluated the efficiency of traffic demultiplexing when defending against various traffic analysis attacks.

Table 2: Network performance: original vs. MD ($I = 3$)

App.	Original ($KBps$, s)		MD ($KBps$, s)	
	Troughput	Delay	Troughput	Delay
br.	55.81	0.0194	53.60	0.0208
do.	296.78	0.0036	264.34	0.0043
up.	216.56	0.0049	209.90	0.0052
vo.	111.30	0.0123	109.96	0.0142
bt.	142.74	0.0083	130.28	0.0087

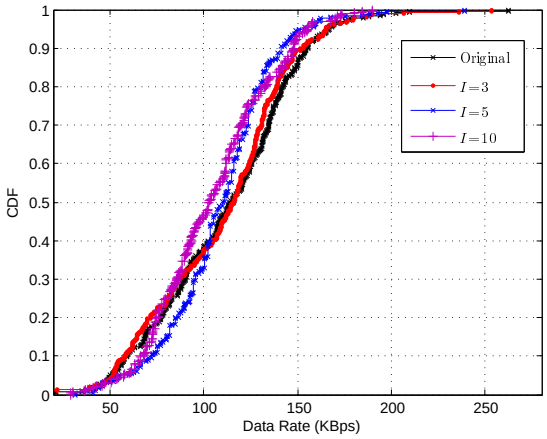


Figure 9: CDF of bandwidth consumption for online video vs. different number of virtual interfaces I.

5.2 Network Performance

5.2.1 Throughput and Delay

In this subsection, we discuss the network performance of traffic demultiplexing. We tested traffic demultiplexing in the testbed with five applications, shown in Table 2. Experiment results showed that the average throughput for the original traffic was a little bit larger than traffic demutliplexing with MD, and the average packet interarrival time of the original traffic was smaller. The small increase in delay and decrease in throughput was because we enabled the promiscuous mode of the wireless card and disabled MAC-layer ACK in order to implement the MAC layer virtualization. In the promiscuous mode, a receiver filters out the packets for other destinations by software instead of hardware. Without MAC-layer ACK, more upper-layer retransmissions will occur to compensate the MAC layer loss. However, by using the traffic demultiplexing technique, the change in throughput and delay is almost unnoticeable. We will show, in next subsection, that traffic demultiplexing does not affect the available bandwidth for other users.

5.2.2 Scalability

We evaluated the scalability of traffic demultiplexing by investigating the performance of a bandwidth-consuming application, "online video," when setting the different number of virtual interfaces (I). Figure 9 shows the CDF of bandwidth consumption when we ran the same online video application on different I. It demonstrates that the bandwidth consumed under different virtual interfaces remains almost unchanged despite the increase of I. Traffic demultiplexing scales well when we increase the number of virtual interfaces. Hence, the adoption of traffic demultiplexing by a user does not affect the experience of others.

Table 3: Accuracy of classification ($W = 5s$) (%)

App.	Original	Morphing	MD
br.	37.77	31.37	0.26
ch.	77.93	72.15	75.66
ga.	88.18	71.68	37.34
do.	99.88	100.00	99.90
up.	95.92	95.92	91.33
vo.	93.32	91.81	0.00
bt.	89.68	37.54	4.97
Mean	**83.24**	**71.18**	**44.21**
Overhead	**0.00**	**39.44**	**13.53**

5.3 Effectiveness and Efficiency

We investigated the three common attack scenarios to evaluate the efficacy of traffic demultiplexing. Based on the description in Section 4, we assumed that an adversary cannot recognize the difference between virtual and real interfaces.

5.3.1 Against the Identification of Online Activities

We built a classification system by incorporating SVM and NN techniques, to infer a user's online activities among seven online applications: browsing (br.), chatting (ch.), gaming (ga.), downloading(do.), uploading(up.), online video (vo.) and Bit-Torrent (bt.) [7]. Traffic traces were collected as training data to build the classification model. After that, we set an "eavesdropping duration" (denoted as W), which is the shortest time duration for testing data. Characteristics we employed in the classification system were: *number of packets*, *max/min/average/standard deviation of packet size*, and *interarrival times* in downlink and uplink. Our experiments also showed that an adversary could infer what a user is doing with an accuracy of 83.24% in 5 seconds (i.e., $W = 5s$), and the accuracy achieved 91.86% when the eavesdropping duration, W, increased to 60 seconds. The traffic demultiplexing with MD decreased the classification accuracy significantly with a small overhead, as shown in Table 3. The overall accuracy of MD was 44.21%. Furthermore, the accuracy for MD barely rose along with the increase of the eavesdropping duration W. They nearly remained unchanged at 46.29% when eavesdropping duration was set at 60 seconds (Due to the space limit, we do not show the results for $W = 60$ seconds).

In terms of accuracy, we find that gaming, browsing, online video, and BT applications are undetectable in MD. In contrast, the classification accuracy was pretty high for chatting, downloading and uploading. Downloading is even higher than the original case. The reason is that most applications are disguised as a composition of chatting and downloading, hence the classification tends to identifying all the traffic as chatting or downloading. In addition, uploading is the only application which has low traffic in downlink but high traffic in uplink, compared with other applications. Hence, it is hard to disguise uploading as another application without packet splitting.

We compared traffic demultiplexing with traffic morphing as shown in Table 3. Specifically, we morphed chatting to gaming, disguised gaming as browsing, simulated browsing as BT, made BT look like online video, pad video to be downloading. Traffic morphing reduced the accuracy to 71.18%. It had less overhead than packet padding but still with 39.44% overhead. Our defensive method, traffic demultiplexing with MD, achieved lower accuracy of 44.21% with a even smaller overhead, 13.53%. The overhead was introduced to ensure traffic on an individual virtual interface like a real application, described in Section 4.3. These data show that traffic

Table 4: Accuracy comparison for web applications

Strategies	Accuracy (%)	Overhead (%)
Original	30.6	0.0
Traffic Morphing	6.3	9.3
Random Padding	5.3	10.6
MD ($I = 3$)	5.0	1.7

demultiplexing performed better than traffic morphing in our measurements. It was a significant improvement in both privacy and overhead.

5.3.2 Against the Identification of Web Pages

As described in Section 4, an attacker usually cannot infer if a Wi-Fi network adopts the traffic demultiplexing technique, or determine the application used by a user. However, an attacker may guess the application of a specific user and apply the existing traffic analysis attacks on a specific interface where the traffic looks like the assumed application. If an attacker assumes the web browsing applications, we show the efficiency of traffic demultiplexing under the web page identification attack using naive Bayes classifier (NBC) [10, 13].

We first set up the traffic analysis attack to system [1, 10], and used morphing, random padding, and traffic demultiplexing to defend against such an attack, respectively. We collected traffic traces of top 80 web pages of USA according to Alexa [36] in encrypted Wi-Fi networks. Based on packet size in the MAC layer, we investigated the identifiability of these web pages. Table 4 shows the comparison of the performance of traffic demultiplexing with random padding and traffic morphing. For random padding, every packet was appended with a padding of random length in $[0, 256)$ bytes to achieve a good tradeoff between efficiency and overhead. For traffic morphing, we used the same optimal scheduling as [13]. In traffic demultiplexing, the number of virtual interfaces is 3 ($I = 3$). The packet size ranges for MD were $[0, 500)$, $[500, 1000)$, and $[1000, \ell_{max}]$ bytes.

We observed that the traffic demultiplexing achieved the best performance in terms of both accuracy and overhead. Note that the classification accuracy values in our experiment are smaller than those in [13]. It is because the data collected in our experiment is from the MAC layer, while the data in [13] is from the transport layer, so we cannot tell whether the packets we collected belong to one web page. Also, when we implemented the traffic morphing, we did not split packets to reduce overhead, and did not apply it on a data set as large as in [13].

5.3.3 Against the Identification of VoIP Features

In this subsection, we show how traffic demultiplexing defends against the identification of VoIP features. We collected VoIP data for 40 minutes by using a variable bit-rate (VBR) codec such as *speex* [37]. The data are encrypted by the IETF standard Secure Real-time Transport Protocol [38]. We implemented a similar testbed as [4] and tested the differentiation between English and Spanish by using the naive Bayes classifier. We compared the efficiency of traffic demultiplexing with random padding and traffic morphing. For random padding, every packet was appended with a padding of random length in $[0, 60)$ bytes. For traffic morphing, we use the optimal scheduling as similar as [13]. In traffic demultiplexing, the ranges of packet size are $[0, 32)$, $[32, 60)$, and $[60, \ell_{max}]$ for MD. Note that we obtained three accuracy numbers corresponding to the three traces that MD produces. The reported the highest among the three.

From the results shown in Table 4, we see that traffic demultiplexing achieves the best protection against information leak with a significantly reduced overhead.

Table 5: Accuracy comparison for VoIP applications

Strategies	Accuracy (%)	Overhead (%)
Original	78.3	0.0
Traffic Morphing	65.0	36.3
Random Padding	68.3	61.7
MD ($I = 3$)	60.0	11.2

6 Related Work

6.1 Traffic Analysis Attacks and Defenses

Traffic analysis on side-channel information leaks has threatened users' privacy in various of applications, such as SSH [6], keystroke dynamics [39] [40], web browsing [1,2,41], video streaming [3], and VoIP [4,5] etc. Encrypted traffic does not prevent an attacker from violating users' privacy through traffic analysis. Traffic analysis attacks on encrypted HTTP streams [1, 2, 41] are reported to be able to reveal what a user is browsing. The traffic characteristics on encrypted VoIP packets [5] can be used to identify phrases used in a call. In addition, due to the shared-medium nature, user identification, location, tracking, and behaviors are all at risk from state-of-the-art traffic analysis attacks [18, 29, 30, 42, 43].

Regarding the defense mechanism, high-level mitigation policies, such as packet padding [1, 2] and traffic morphing [13], are likely to be either inefficient or incur significant overhead. Pseudonyms [17, 18] may still lead to information leaks in terms of coarse granularity of traffic partition. The identifier-free approach proposed in [16] encrypts all explicit identifiers from all transmitted bits to improve privacy, but the overhead of encryption and key management cannot be overlooked. Hence, an efficient and effective defense against side-channel information leaks is an important research topic with strong practical relevance.

6.2 Virtualization in Wi-Fi Networks

Recently,researchers have introduced virtualization into Wi-Fi networks. A fat virtual AP introduced in [44] is an 802.11 driver that aggregates the bandwidth available at accessible APs for users and also balances their loads. VirtualWiFi [45] or MultiNet [46] uses a network hopping scheme to switch the wireless card across multiple APs of wireless LANs with one MAC address. The PeerBoost [47] system uses the coexistence of infrastructure and ad-hoc modes over the one wireless card to maximize the throughput of Wi-Fi networks. Different from previous research, our work on traffic demultiplexing creates multiple virtual interfaces and shapes different traffic characteristics on individual interfaces to protect user privacy.

7 Conclusions and Future Work

In this paper, we propose *traffic demultiplexing* to protect user privacy. It creates multiple virtual MAC interfaces, dynamically dispatches traffic flows over these interfaces, and rebuilds different traffic characteristics for each virtual interface to obscure the original patterns. Since traffic demultiplexing does not require aggressively padding packets or adding cover traffic, and therefore avoids substantial communication overhead. In addition, traffic demultiplexing is transparent to upper layers and does not change the user experience. We evaluate the performance of traffic demultiplexing through real experiments in WLAN testbeds. Experimental results demonstrate the effectiveness and efficiency of traffic demultiplexing in defending against various traffic analysis attacks. Further, traffic demultiplexing has good scalability and hence is suitable for WLANs deployed in residential, hotspot, and campus environments.

Although we implement traffic demultiplexing by modifying the MadWifi driver in our testbed prototype, the defensive approach we proposed is more general and can be adopted in many other wireless device drivers, such as *ath5k* and *iwlagn*. In addition, it can be applied to other network layers for the general purpose of privacy enhancement. For example, we can adopt traffic demultiplexing in the IP layer to defend against traffic analysis in Ethernet. We believe that traffic demultiplexing can help relieve today's application related privacy concerns in general and effectively enhance privacy-preserving in wireless networks in particular.

The discussion of this paper mainly focuses on a passive adversary who eavesdrops on the Wi-Fi channel. In a more powerful threat model, the adversary can even actively probe MAC interfaces through sending messages to the victim. This can happen, for example, when the victim has an email client running or uses a P2P software. In this case, the adversary could fingerprint the feature of the packet she send (e.g., a very special size of an email) to infer the victim's virtual interfaces by checking which interface receives the packet. This type of traffic marking techniques has been discussed in prior research [48, 49]. Such a threat can be greatly mitigated by tieing a set of virtual interfaces to a specific online activity that needs protection. For example, we can use three interfaces to cover the web content the user is browsing, while reserving another interface for less sensitive activities such as receiving emails and performing P2P communications. As a result, the adversary capable of contacting the user through these activities will not be able to link the user to the three interfaces. In our follow-up research, we will further study this threat and other possible countermeasures.

References

[1] Q. Sun, D.R. Simon, Y. Wang, W. Russell, V.N. Padmanabhan, and L. Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings of IEEE Symposium on Security and Privacy*, 2002.

[2] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 191–206, 2010.

[3] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *Proceedings of USENIX Security Symposium*, 2007.

[4] C.V. Wright, L. Ballard, F. Monrose, and G. M. Masson. Language identification of encrypted VoIP traffic: Alejandra y roberto or alice and bob. In *Proceedings of USENIX Security Symposium*, 2007.

[5] C.V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson. Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations. In *Proceedings of IEEE Symposium on Security and Privacy*, 2008.

[6] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. ACAS: automated construction of application signatures. In *Proceedings of the 2005 ACM SIGCOMM workshop on mining network data*, pages 197–202. ACM, 2005.

[7] F. Zhang, W. He, X. Liu, and P. Bridges. Inferring users' online activities through traffic classification. In *Proceedings of WiSec*, 2011.

[8] A. Dainotti, W.D. Donato, A. Pescape, S. Rossi, et al. Classification of network traffic via packet-level hidden Markov models. In *Proceedings of GLOBECOM*, pages 1–5. IEEE, 2008.

[9] C.V. Wright, F. Monrose, and G. Masson. HMM Profiles for Network Traffic Classification (Extended Abstract). In *Proceedings of Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC)*. Citeseer, 2004.

[10] M. Liberatore and B. Levine. Inferring the source of encrypted http connections. In *Proceedings of Computer and Communications Security*, 2006.

[11] Google says it inadvertently collected personal data, 2010. http://bits.blogs.nytimes.com/2010/05/14/ google-admits-to-snooping-on-personal-data/.

[12] RFC4949, August, 2007. Internet Security Glossary, Version 2.

[13] C.V. Wright, S.E. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *Proceedings of NDSS*, 2009.

[14] The Tor Project. Tor: anonymity online. http://www. torproject.org/.

[15] P. Golle, X. Wang, M. Jakobsson, and A. Tsow. Deterring voluntary trace disclosure in re-encryption mix networks. In *Proceedings of IEEE Symposium on Security and Privacy*, 2006.

[16] B. Greenstein, D. Mccoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *Proceeding of MobiSys*, 2008.

[17] M. Gruteser and D. Grunwald. Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. *ACM Mobile Networks and Applications*, 10(3):315–325, 2005.

[18] T. Jiang, H.J. Wang, and Y. Hu. Preserving location privacy in wireless LANs. In *Proceedings of MobiSys*, pages 246–257, 2007.

[19] B. Greenstein D. Grunwald K. Bauer, D. McCoy and D. Sicker. Physical layer attacks on unlinkability in wireless LANs. In *Proceedings of PETS*, 2009.

[20] F. Zhang, W. He, and X. Liu. Defending against traffic analysis in wireless networks through traffic reshaping. In *Proceedings of ICDCS*, 2011.

[21] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet traffic classification demystified: myths, caveats, and the best practices. In *Proceedings of the 2008 ACM CoNEXT conference*, pages 1–12. ACM, 2008.

[22] A. Moore and D. Zuev. Internet traffic classification using Bayesian analysis techniques. *ACM SIGMETRICS Performance Evaluation Review*, 33(1):50–60, 2005.

[23] S. Lakshmanan, C.L. Tsao, R. Sivakumar, and K. Sundaresan. Securing wireless data networks against eavesdropping using smart antennas. In *Proceedings of ICDCS*, 2008.

[24] I. Martinovic, P. Pichota, and J.B. Schmitt. Jamming for good: design and analysis of a crypto-less protection for WSNs. In *Proceedings of WiSec*, 2009.

[25] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T.T. Abdelzaher. Pda: Privacy-preserving data aggregation in wireless sensor networks. In *Proceedings of INFOCOM*, pages 2045–2053. IEEE, 2007.

[26] T. Li, N. Li, J. Zhang, and I. Molloy. Slicing: A New Approach to Privacy Preserving Data Publishing. *Arxiv preprint arXiv:0909.2290*, 2009.

[27] M. Strasser, S. Capkun, C. Popper, and M. Cagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *Proceedings of Security and Privacy*, pages 64–78. IEEE, 2008.

[28] http://madwifi-project.org/.

[29] J. Wilson and N. Patwari. See through walls: Motion tracking using variance-based radio tomography networks. *IEEE Transactions on Mobile Computing*, 2010.

[30] P. Tao, A. Rudys, A. M. Ladd, and D. S. Wallach. Wireless LAN location-sensing for security applications. In *Proceedings of WiSE*, 2003.

[31] K. Bauer, D. McCoy, B. Greenstein, D. Grunwald, and D. Sicker. Performing traffic analysis on a wireless identifier-free link layer. In *Proceedings of Richard Tapia Celebration of Diversity in Computing*, pages 18–23, 2009.

[32] K. Kowalik, M. Bykowski, B. Keegan, and M. Davis. Practical issues of power control in IEEE 802.11 wireless devices. In *Proceedings of International Conference on Telecommunications*, pages 1–5. IEEE, 2008.

[33] K. Ramachandran, R. Kokku, H. Zhang, and M. Gruteser. Symphony: synchronous two-phase rate and power control in 802.11 WLANs. In *Proceeding of MobiSys*, pages 132–145. ACM, 2008.

[34] hostapd. http://hostap.epitest.fi/hostapd/.

[35] Libpcap. http://www.tcpdump.org/.

[36] Top sites in united states. http://www.alexa.com/topsites/countries/US.

[37] Speex. http://www.speex.org/.

[38] M. Baugher, D. McGrew, M. Naslund, and E. Carrara. The secure real-time transport protocol (SRTP).

[39] F. Monrose and A. Rubin. Authentication via keystroke dynamics. In *Proceedings of Computer and Communications Security*, pages 48–56, 1997.

[40] X. Song, D. Wagner, S. David, and X. Tian. Timing analysis of keystrokes and timing attacks on SSH. In *Proceedings of USENIX Security Symposium*, 2001.

[41] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine. Privacy vulnerabilities in encrypted HTTP streams. In *Proceedings of Privacy Enhancing Technologies Workshop*, pages 1–11, 2005.

[42] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *Proceedings of MobiCom*, pages 99–110. ACM Press, 2007.

[43] J. Franklin, D. Mccoy, P. Tabriz, and V. Neagoe. Passive data link layer 802.11 wireless device driver fingerprinting. In *Proceedings of USENIX Security Symposium*, pages 167–178. USENIX Association, 2006.

[44] S. Kandula, K. C. Lin, T. Badirkhanli, and D. Katabi. Fatvap: Aggregating ap backhaul capacity to maximize throughput. In *Proceedings of NSDI*, 2008.

[45] http://research.microsoft.com/en-us/projects/virtualwifi/.

[46] R. Chandra and P. Bahl. MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card. In *Proceedings of INFOCOM*, volume 2, pages 882–893. IEEE, 2004.

[47] O. Barak, R. Friedman, and G. Kliot. PeerBooster: Enhancing Throughput in Wi-Fi Networks Through Network Virtualization.

[48] X. Wang, S. Chen, and S. Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2007.

[49] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao. Dsss-based flow marking technique for invisible traceback. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 18–32, 2007.