RAVEN: Stateless Rapid IP Address Variation for Enterprise Networks

Liang Wang
Princeton University
lw19@princeton.edu

Hyojoon Kim
Princeton University
hyojoonk@cs.princeton.edu

Prateek Mittal
Princeton University
pmittal@princeton.edu

Jennifer Rexford
Princeton University
jrex@cs.princeton.edu

ABSTRACT

Enterprise networks face increasing threats against the privacy of their clients. Existing enterprise services like Network Address Translation (NAT) offer limited privacy protection, at the cost of requiring per-flow state. In this paper, we introduce RAVEN (Rapid Address Variation for Enterprise Networks), a network-based privacy solution that is *complementary* to application-layer defenses. RAVEN protects privacy by frequently changing the client's public IP address. With RAVEN, a client is not limited to using a single IP address at a given time, or even for a given connection. RAVEN goes further, breaking the association between packets that belong to the same connection by frequently changing the client's IP address *within* a single connection. RAVEN achieves this through a novel division of labor: the client uses a transport protocol, like QUIC, that supports seamless connection migration, and decides *when* to switch its IP address, while the enterprise network actually changes the client's IP address in a stateless manner at line rate and ensures end-to-end packet delivery. We implement RAVEN using QUIC and off-the-shelf programmable switches. We deploy RAVEN in a test IPv6 network and evaluate its defense against webpage fingerprinting attacks. Even with a strong adversary, the average precision of the best adaptive attacks drops from 0.96 to 0.84, with a 0.5% degradation in client throughput. When RAVEN changes IP addresses at unpredictable frequency, the precision of the best attacks falls to 0.78—the same effectiveness as WTF-PAD.

KEYWORDS

privacy, traffic analysis, programmable data plane, P4, QUIC

1 INTRODUCTION

Clients in an enterprise network (e.g., campus, corporate, or access network) are exposed to many attacks that threaten their privacy in today's connected world. For example, an Internet Protocol (IP) address can be used to identify a client and device communicating on the Internet, enabling client activity and location tracking [13, 27, 42, 48, 53, 69–71, 73]. The problem will become more severe as enterprise networks adopt IPv6, where each client may get a persistent, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [3], enabling Apple device users to protect their privacy by directing their traffic through two or more private relays. However, Private Relay cannot be applied to clients that do not allow much customization or control, such as Internet-of-Things devices.

Can an enterprise network play a more active role and provide a fundamental solution that is less client- or device-dependent? A common practice is Network Address Translation (NAT) which can change IPs per client (or even per flow). Address Hiding Protocol (AHP) [49] and Lightweight Anonymity and Privacy (LAP) [32] improve on NAT by encrypting the IP (instead of mapping) per flow in more efficient ways. However, breaking the association between the flows belonging to the same client is not enough. In fact, many advanced traffic-analysis attacks remain viable by grouping enough packets of a *single* connection together. For instance, adversaries can also find out what specific webpages the clients visit (i.e., webpage fingerprinting) by analyzing traffic patterns [28, 31, 47, 50, 57, 67], even if the server IP addresses appear to be the same to the adversaries. A corporate enterprise, for example, would want to prevent adversaries from fingerprinting which webpages the company's employees are accessing, to avoid leaking sensitive information about its current business. As another example, an access network provider would want to prevent adversaries from inferring that the customers in a geographic region belong to a particular minority group based on the webpages they disproportionately access.

An even stronger mitigation strategy is to break the association between *the packets belonging to the same flow or connection.* This not only prevents an adversary from learning about clients, but also makes it learn less information from a connection, which can thwart various traffic-analysis attacks. MIMIQ [25] tries to realize this idea using QUIC's connection migration feature, but does not provide a complete system (see § 2.3 for more details). Inspired by MIMIQ, we present RAVEN (Rapid Address Variation for Enterprise Networks), a new privacy primitive for an enterprise network to protect all of its clients. RAVEN is a network-based solution that can run independently, or be combined with application-layer solutions. RAVEN improves on existing enterprise privacy solutions (e.g., NAT and MIMIQ) by providing packet-level unlinkability in a stateless way. RAVEN uses a unique technique for breaking the link between transport connections (or flows) and clients: **rapid**

Proceedings on Privacy Enhancing Technologies 2023(3), 1–17
© 2023 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2023-0066

address variation. The core idea is to rapidly rotate a client's public metadata fields in packets, such as an IP address, in an unpredictable way to the adversary, *even within a single flow*. Such *ephemeral* IP addresses would leave a short time window for the adversary, making it difficult to associate a flow to a client or to analyze traffic content.

However, such a mechanism is hard to realize either by the client or the network alone. First of all, if the *client* rotates its IP address without the network's support, end-to-end packet delivery is no longer guaranteed. Secondly, if the *network* rotates the client's IP address rapidly, it might break the client's connectivity with conventional Internet services, which would dramatically degrade performance and user experience. To address this, RAVEN proposes a new division of labor between clients and the network. In RAVEN the client decides *when* to change IP addresses, while the network actually changes the IP addresses and ensures that return traffic is delivered. This enables the client to strike its own balance between privacy and performance, while the network ensures efficient end-to-end packet delivery.

By refactoring the relationship between clients and the network, RAVEN can be implemented and deployed in practice with good performance. The time is ripe, with three emerging trends coming together to enable our system:

- **Servers that allow client IP addresses to change:** Transport protocols like *QUIC* [35] have a connection migration capability—initially designed for seamless mobility—that allows the client's IP address to change within the life of a connection.
- **Network devices that can encrypt and decrypt IP addresses:** *Programmable data planes* parse and manipulate packets at line rate, allowing an enterprise network to encrypt and decrypt IP addresses at the packet level.
- **Core networks that support large IPv6 addresses:** Many transit networks already deliver *IPv6* traffic. The 128-bit IPv6 addresses give us plenty of room to encode an encrypted client IP address.

Using encryption and decryption for IP address translation, RAVEN does not require the switches to maintain per-connection state. The **stateless** design makes RAVEN naturally robust to asymmetric routing and routing changes. In addition, RAVEN obviates the need for the enterprise to run a NAT or stateful firewall, by having the border switches translate client IP addresses and detect (and drop) unsolicited server traffic.

We implement RAVEN using the QUIC protocol [24] and an off-the-shelf Intel Tofino switch [34]. RAVEN can be deployed incrementally. For an unmodified client and server, the RAVEN switch can rotate the client IP address per-connection or even every few packets. To further enhance privacy, we modify the QUIC client to rotate *transport-layer* metadata, such as port numbers and QUIC connection IDs, which could otherwise be used to group related packets in a traffic-analysis attack. With a small modification to the QUIC server software, we can avoid the server unnecessarily resetting the connection's congestion-control state after an IP rotation and path validation, leading to improved performance.

We deploy RAVEN in a publicly-reachable /64 network testbed. We validate that RAVEN can successfully perform client IP rotation at a high frequency to thwart IP-based attacks. We evaluate state-of-the-art webpage fingerprinting (WF) attacks against RAVEN under

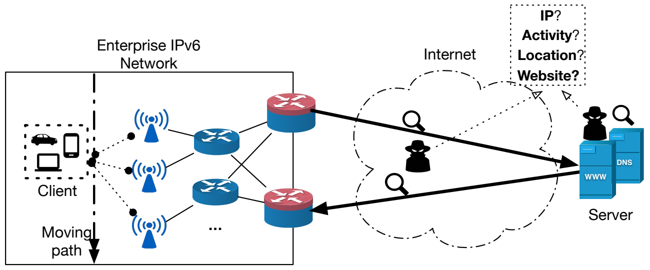

Figure 1: Enterprise network and privacy threats.

adaptive attack strategies. [1] Even with a strong adaptive adversary that could identify a large portion of the packets in a protected connection, the (average) precision of the best-performing WF attacks drops from 0.96 to 0.84, with a 0.5% client throughput degradation. By changing the migration strategy, RAVEN can achieve similar efficiency as an application-layer WF defense WTF-PAD [38]. Combining RAVEN with WTF-PAD, the precision drops further to 0.64. When used alone, RAVEN can also reduce the precision of the best adaptive attacks down to 0.65, with a cost of 20% throughput degradation.

The dataset for WF evaluation contains 12,000 traces, which were collected by repeatedly visiting 100 monitored webpages twenty times and 10,000 unmonitored webpages on GitHub Pages, and the traffic was split by RAVEN. We make this dataset and RAVEN source code publicly available [63].

Ethics statement: In our RAVEN deployment, only synthetic clients that we control use RAVEN. We analyzed packet traces from MAWI [12], which are publicly available and anonymized. We also analyzed network traffic captured on a large campus network. A network operator has inspected and sanitized all packet traces to remove all personal data and payload before being accessed by researchers. We have acquired all necessary approvals from the institute beforehand, including its Institutional Review Board (IRB).

2 RAVEN PROBLEM SETTING

As shown in Figure 1, we consider an enterprise IPv6 network that has the incentives to protect client privacy. It could be a corporation that wants to hide which web resources the employees are accessing from adversaries outside of the network to protect sensitive information, or a network provider that provides a value-added service to privacy-savvy users. Clients may or may not fully trust the organization running the network, and can use client-based anonymity techniques (e.g., Tor and VPNs) to hide destinations from the network. However, the clients, who want to fully leverage the privacy benefits provided by the network, are still willing (or required in certain settings) to make necessary changes to cooperate with the network. Clients can be stationary or mobile across different subnets. The network may have multiple border switches, subject to asymmetric routing and routing changes; in particular, the outgoing and incoming traffic could go through different border switches at different times.

[1]RAVEN may not work effectively against website fingerprinting because RAVEN does not hide server IPs (and ports). It could be trivial to distinguish different websites based on server IPs. In webpage fingerprinting, all webpages share the same server IP.

In this section, we first present our threat model and design goals, followed by a comparison of RAVEN with existing solutions.

2.1 Threat model

We consider two types of *passive* adversaries: (**A1**) on-path eavesdroppers (i.e., any network element between the network and the server, such as an AS or an Internet exchange point) and (**A2**) malicious servers that clients communicate with. *We do not consider attacks that leverage non-public information such as a username, password, or cookies.* Depending on their capabilities, adversaries are able to perform different attacks.

IP-based fingerprinting and tracking attacks (A1, A2) A client's IP address could serve as a pseudonym for the client, even behind a NAT [42]. Analyzing the traffic pattern between the client and servers may allow an eavesdropper (**A1**) to fingerprint the client or infer client activity [48]. The adversary may also discover compromised clients or hosts (by worms, viruses, etc.) based on traffic patterns [69, 70] and select targets for future attacks. An **A2** adversary can profile clients and track their activity only based on IP [42, 71, 73], even without cookies or when privacy browsing is being used.

Another threat is *location tracking*. Though geopositioning based on IPv4 addresses sometimes has poor accuracy [55], the situation may change in IPv6. RIPE's best current practice for IPv6 prefix assignment [52] suggests that customer premise equipment, such as an access point, is able to get a dedicated, up to /64 IPv6 prefix. Such a prefix can be treated as a location ID, and an adversary can geolocate a client with high accuracy with auxiliary data from location services (e.g., the client is attached to a certain access point or subnet within a network). Furthermore, the lowest 64 bits of an IPv6 address may remain constant even when the client moves among different networks, as it could be generated based on the client's MAC address by Stateless Address Auto Configuration (SLAAC) [62]. This allows an adversary to pinpoint the victim's locations and also track the client moving across different networks [13, 27, 53].

Traffic-analysis attacks on the enterprise network (A1). A variety of traffic-analysis attacks leverage statistics extracted from network flows and do not need to know the true IP address of each client. An adversary may perform traffic-analysis attacks to learn sensitive information about the client's enterprise network. For example, the adversary may try to figure out the web page/resource of interest to the majority of clients during a time period. Obfuscating an IP address and port number is not sufficient for such fingerprinting attacks [28, 31, 47, 50, 57, 67].

Specially, we focus on webpage fingerprinting attacks in the paper in which the resources to fingerprint are behind the same IPs, e.g., resources on the CDNs. An analysis of campus traffic (Table 1) suggests that major CDNs support QUIC and a single CDN IP address can be associated with multiple concurrent connections, providing a reasonable anonymity set. § 6.4 provides a more detailed analysis of the anonymity set.

2.2 Design goals

Motivated by our threat model, we want to design a system that achieves the following privacy properties in an enterprise network setting.

P1. Lightweight sender anonymity. An adversary outside the enterprise network cannot discover the IP address of the client (sender). Note that we do not try to hide the Autonomous System (AS) of the client. This is the same goal as AHP [49].

P2. Flow unlinkability. Here, we define flow as a connection in TCP or UDP. We define flow unlinkability as the following: given a set of flows, the adversary cannot determine whether the flows are associated with the same client based on observed public metadata. This property helps protect clients against tracking attacks that exploit the information from multiple flows, such as IP-based webpage identification attacks proposed by Patil and Borisov [48].

P3. Subflow unlinkability. In this work, we propose a novel privacy property called *subflow unlinkability*. With subflow unlinkability, given segments of a flow (i.e., subflows), the adversary cannot reliably determine whether the subflows belong to the same flow or are associated with the same client based on the observed public metadata. Much stronger than flow unlinkability, fulfilling this property enables a more robust defense against advanced traffic-analysis attacks, as it *restricts the information the adversary can learn from traffic*. Though prior work on traffic-splitting-based webpage fingerprinting defenses [29, 39, 65] achieves this property to some extent, it has not been fully conceptualized in the context of webpage fingerprinting. We demonstrate the effectiveness of this property against webpage fingerprinting attacks in § 6.

We also set several key operational (**O**) goals:

O1. No connectivity disruption. The rotation of metadata, particularly IP addresses, shall not disrupt ongoing connections. Clients shall not suffer from packet drops, retransmissions, or connection reestablishment during metadata rotation.

O2. No client application modification. For greater compatibility, the solution shall not require modifications to existing applications used by the client.

O3. Robust to routing changes and asymmetry. Outgoing traffic and return traffic may take different routing paths, and these paths may change over time. The solution shall gracefully handle asymmetric routing and routing changes.

O4. Minimal performance impact. The solution shall incur minimal performance degradation, like reduced throughput or increased latency, for the client.

O5. Low deployment and maintenance overhead. The solution shall minimize the deployment and maintenance overhead, like extra state (e.g., stateful NAT) or servers (e.g., VPN) required for running a solution.

2.3 Comparison with existing solutions

It is challenging to achieve these privacy properties while still fulfilling the operational goals. We make this point by discussing how existing enterprise solutions work (Table 2).

NAT. Although not its primary goal, a NAT device could provide some privacy benefits: sender anonymity and flow unlinkability (with NAT oversubscription). However, NATs cannot provide subflow unlinkability (**P3**) without disrupting ongoing connections. Also, NAT devices need to maintain state (**O5**); and syncing connection state between multiple NATs at the border is a challenge for **O3**.

ASN	Provider	# Flows (%)	# Server IP
AS15169	Google	27,264,590 (81.37)	7,538
AS32934	Facebook	1,926,463 (5.75)	356
AS8075	Microsoft	1,863,316 (5.56)	649
AS13335	Cloudflare	908,982 (2.71)	18,252
AS36183	Akamai	726,661 (2.17)	110
AS396982	Google Cloud	601,520 (1.8)	1,304
AS16509	Amazon (CloudFront)	68,144 (0.20)	3,794
AS54113	Fastly	45,003 (0.13)	220
AS714	Apple	30,255 (0.09)	302
AS20940	Akamai	8,816 (0.03)	803

Table 1: Top 10 destination ASes associated with the most QUIC connections. A total of 33.5 M QUIC connections were observed in two weeks. Note that more than 99% of the observed destination IPs belonging to AS16509 are used by CloudFront.

	P1	P2	P3	O1	O2	O3	O4	O5
NAT	✓	✓*		✓*	✓		✓	
Standard IP reassign	✓			✓	✓	✓	✓	✓
AHP	✓	✓		✓	✓	✓		
MIMIQ	✓	✓		✓	✓	✓		
RAVEN	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: Comparison of existing solutions."*" indicates the goal is achieved conditionally depending on solutions.

Standard IP reassign. An enterprise network may seek to reassign IPv6 addresses to clients via SLAAC very frequently to achieve **P1**, **P2**, or even **P3**. Indeed, several practices are recommended for IPv6 (RFC 4941 [44], prefix rotation [53], and others [13, 19–21]), which focus on generating frequently-changing "temporary" IPv6 addresses. However, to avoid disrupting ongoing connections, the rotation frequency is still low—typically 24 hours or more, which undermines the potential privacy benefits. Besides, such solutions do not even achieve flow unlinkability (**P2**) in practice because a client only gets a single IP address at a time.

AHP. In Address Hiding Protocol (AHP) [49], a trusted network changes a client's IP address to another one in the network's own IPv4 address space through encryption. This conceals the client's identity without any client participation. Different flows have different source IP addresses, thus AHP can provide sender anonymity and flow unlinkability. However, AHP is not designed to provide subflow unlinkability (**P3**). Moreover, AHP performs expensive cryptographic operations that are difficult to run at line rate in the network, and needs to maintain flow state to avoid breaking ongoing TCP connections, which makes it difficult to scale to IPv6 networks (**O4, O5**).

MIMIQ. Though MIMIQ [25] pioneered the idea of leveraging QUIC connection migration to hide client IP addresses, it does not provide subflow unlinkability (**P3**): MIMIQ did not change the associated connection ID field, which could be used to trivially link multiple packets of the same connection together. MIMIQ also relied on a DHCP-like allocation server to remember the IP address mapping, introducing state (**O5**). When migrating at a high frequency, the client throughput drops significantly in MIMIQ (**O4**).

We discuss other network-based privacy solutions (e.g., network-layer anonymity systems) and client-based traffic splitting in § 8.

Key takeaways:

- To achieve subflow unlinkability (**P3**), we need to change the source IP address of packets *within* a flow. Yet, this disrupts the ongoing connectivity (i.e., disrupts **O1**). A promising approach is to leverage protocols that support mobility, TCP Migrate, MPTCP, QUIC and WireGuard, etc., as demonstrated in prior work [25, 39, 58, 64, 65]. However, we must avoid privacy leakage from non-IP identifiers, which are used in these protocols as an alternative for IP address (e.g., QUIC's connection ID).
- **P2** requires the client to (a) use different IPs for different flows, or (b) in the reverse direction all flows share the same set of client IPs, like in NAT.

3 RAVEN DESIGN

RAVEN leverages the division of labor between the client transport protocol and the network switches to rotate public traffic metadata (§ 3.1). In RAVEN, a modified transport protocol frequently rotates transport-layer metadata, like QUIC's connection ID (§ 3.2), while the network reactively generates encrypted IP addresses and port numbers on-the-fly and assigns them to packets (§ 3.3).

3.1 RAVEN's division of labor

With a *client*-oriented approach, the network assigns multiple IP addresses to a client and lets the client proactively rotate the IP address, port number, and public identifiers in a connection. However, this requires a special client setup and does not scale with a client that has a large number of connections. With a *network*-oriented approach, the network controls the metadata rotation on behalf of the client. In this case, IP rotation is easy via NAT-like techniques. Yet, transport-layer identifier rotation is difficult to achieve as the network has no visibility into the identifier management due to end-to-end encryption. Therefore, our design chooses a middle ground between the two. In RAVEN, the client takes charge of transport-layer metadata rotation without worrying about the IP layer, and the network manages IP addresses and port numbers based on public information in packets without peeking into the encrypted data. Labor is divided to minimize the required modifications to the system's components and also to improve performance.

3.2 Clients with seamless migration

RAVEN leverages the rising popularity of transport protocols like QUIC that support connection migration. Using such transport protocols, remote Internet services allow the client's IP address to change even within the life of a connection. With RAVEN, clients using these transport protocols (instead of TCP) can automatically gain enhanced privacy. While we focus on QUIC as our transport protocol, we note that there can be many alternatives. We chose QUIC as it has been standardized and is the foundation of HTTP3 [35]. We also expect to see widespread adoption of QUIC going forward.

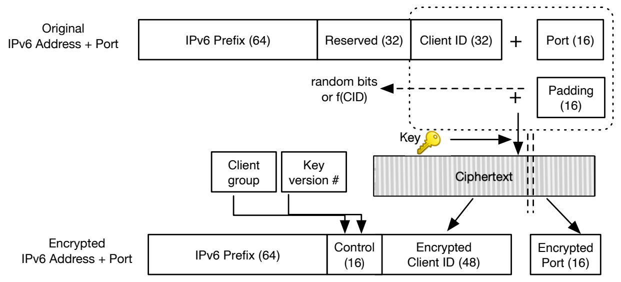

Figure 2: RAVEN IPv6 source address encoding.

Privacy-enhanced QUIC that rotates the connection ID. Yet, unmodified QUIC is still vulnerable to attacks that link packets to a connection, even when an IP address cannot tie packets to a connection. In particular, QUIC uses a *connection ID* (CID) to locate the per-connection decryption keys for further payload processing upon receiving a packet. Packets from the same connection usually share the same CID, which breaks subflow unlinkability (**P3**).

In RAVEN, we extend QUIC so that a connection can actively rotate connection IDs frequently based on fine-grained metrics (e.g., packet counter). The client and the server negotiate a set of CIDs and periodically update the valid CID set. When setting the CID in a packet (or a group of packets), the client selects a random CID from the valid CID set (with or without reusing). CIDs are generated and exchanged in an encrypted way, so a passive adversary cannot learn the set of valid CIDs. By using a randomly-selected CID among a set, and even rotating the set of valid CIDs periodically, the CID itself can no longer be used for identifying all packets that belong to the same connection. This lays the foundation for **P3**.

The client controls the rotation timing: The client is in complete control of *how often* to change its transport layer's connection ID and *when* to change it. For example, a client can even change the CID at gaps in the stream of packets, defending against an adversary that tries to identify and group packets with small inter-arrival times as part of the same connection. Of course, the client has to strike a good balance between privacy and performance.

3.3 A network that encrypts and encodes

In this section, we explain how the network encrypts client IPs. To simplify our discussion, we consider a simple, flat IPv6 network with no subnets. As in Figure 2, the network adopts a special addressing scheme: the non-prefix bits of an IPv6 address assigned to a client is divided into (i) reserved bits and (ii) the client bits (or client ID). The client bits are used for identifying each client while the reserved bits can be any fixed value. For the sake of discussion, we will assume a 64-bit network prefix, a 32-bit reserved, and a 32-bit client ID in our running example.

Encryption using the client ID, port number, and padding bits. First, RAVEN needs to hide the raw network and transport layer metadata in a client's packet, i.e., the client's raw IP address and port number. To achieve this, a RAVEN switch first extracts the client ID bits (from the original client's IP address) and the client's source port. Then a special padding is added, which plays an important role later in rotating the address and port number. After that, the RAVEN switch performs symmetric encryption, which produces an encrypted ciphertext. The last 16 bits of this ciphertext are used as the encrypted port while the remaining bits are encoded into the new encrypted IP address, acting as a unique (but temporary) identifier for the client. For now, we defer the discussion about our encryption scheme until § 3.4.

Instead of encryption, one may hide IPs by mapping an IP address to another address in the same address space. Such a mechanism shares the same limitations as NATs: it requires maintenance of state which makes it difficult to scale to large IPv6 networks and handle routing changes and asymmetric routing.

Rapid rotation of the ciphertext. RAVEN's main goal is to break the association between the packets belonging to the same connection. Thus, the network and transport layer metadata in a packet, i.e., the client's IP address and port number, need to be rotated rapidly *even after they are encrypted.* This is where the padding becomes important. RAVEN supports multiple ways to generate the padding, and we introduce two here:

(1) *Random generation per packet.* In this case, every packet has a different combination of source IP address and port number.
(2) *As a function of connection ID.* In other words, *padding* $= f(CID)$. In this case, the padding would change each time the connection ID changes.

Option (1) is ideal for connectionless applications, like DNS and NTP (see further discussions in § A.6), while option (2) is better for connection-oriented applications running over QUIC, like YouTube streaming with a Chrome browser. By taking the connection ID (CID) as part of the encryption input, the resulting encrypted source IP and port number change accordingly as the CID changes. This is where RAVEN's division of labor is highlighted: the client's transport-layer metadata rotation *directs* the rotation of the public IP address and source port number assigned to the client. Meanwhile, it is the network that actually changes the public IP address and source port number in the client's packet.

The public IP address and port number of a client observed on the Internet change frequently, achieving sender anonymity (**P1**). Different connections will have different source IP addresses because of the difference in CIDs, achieving flow unlinkability (**P2**). Lastly, the rotation of connection ID even within the life of a connection will split a connection into seemingly-independent connections, achieving subflow unlinkability (**P3**).

Stateless cipher operations. All the ciphertext bits are stored in the source IP and port number, so RAVEN does not need to maintain any extra state for decryption. In addition, all RAVEN switches deployed in the network use the same set of keys to perform encryption and decryption as packets traverse the switches. Thus, with RAVEN, the switches at the border do not have to continuously maintain or synchronize per-client or per-connection state among them, which is a significant overhead for stateful firewalls and NAT devices.

Dynamic key rotation and assignment. In our running example in Figure 2, note that there are 16 bits remaining in the encrypted IP address, denoted as *control bits.* We utilize these bits to further enhance security:

(1) Key rotation to limit the number of encryptions per key. Encryption keys are updated periodically, say every t seconds, to limit the

number of plaintext-ciphertext pairs an adversary can collect for each key. Key rotation reduces the attack success probability and minimizes the damage caused by a compromised key. One issue caused by key rotation is inconsistent keys during encryption and decryption, i.e., the key may be updated when the return packets are still in transit. To address this issue, we maintain multiple versions of a key, and rotate the key sets in the same way as SPINE [14]. Without keeping a record of the keys used over time, the network will not be able to decrypt older traffic, which gives the network plausible deniability.

(2) Client group to isolate client traffic. RAVEN can group the clients based on certain criteria and use a different key for each client group. This has two potential benefits. First, if a key gets compromised for whatever reason, it will only impact the client group that was using that particular key. Second, it is possible to run differentiated services for clients by having a different key rotation rate per client group. Each group has a group ID, which will be used for locating the corresponding encryption keys. Similar to the key version, the group ID can also be stored in the control bits. Currently, control bits are only used for identifying three types of clients (§ 3.5).

Both the key rotation and assignment per client group in the control bits can be orchestrated by a logically centralized controller that interacts with all participating switches.

IPv6 metadata obfuscation. Certain IPv6 header fields may serve as flow identifiers to help the adversary reconstruct original flows. One is flow label, a 20-bit random string in the IPv6 header for traffic classification [1, 15]. Another one is the IPv6 flexible extension headers, which may contain much auxiliary information [15]. Similar to the TLS ciphersuite, the combination of extension headers may serve as a "fingerprint" to identify a flow. The cooperative client will use zero-value flow labels and fixed extension headers. In fact, zero-value flow labels are very common, and extension headers are rarely used in IPv6. We present a detailed measurement on the usage of the two fields, and discuss other potential identifiers (QUIC packet number, IP ID, etc.) in Appendix § A.1.

IP address collision. In RAVEN, client IP addresses (port numbers) encrypted by the same key will not have collisions because the encryption is a pseudorandom permutation (1-to-1 mapping). The active keys all have different version numbers that will be set in the reversed bits, which can ensure the addresses encrypted by different keys are different.

Extensions of RAVEN. We have discussed the scenario that the IPv6 network has no subnets. RAVEN can also be applied to protect hierarchical IPv6 addresses from leaking internal routing details, and be extended to protect IPv4 networks or other protocols such as DNS and WireGuard. See more details in § A.6.

3.4 Efficient encryption in the data plane

To achieve the operational goals in § 2.2, RAVEN leverages high-speed programmable switches to perform encryption at line rate. Commodity programmable switches can process terabytes of traffic per second (e.g., Tofino [34]). RAVEN uses the two-round Even-Mansour (2EM) encryption developed in PINOT [64] for line-rate encryption. With 2EM, RAVEN can encrypt a packet in a **single** pass through the packet-processing pipeline. We refer the interested readers to [7, 64] for security analysis on 2EM. Using next-generation programmable switches, RAVEN could also run standard lightweight ciphers, e.g., SIMON/SPECK, that support 64-bit encryption [5].

	Unmod. QUIC	QUIC Client: Modified QUIC Server: Unmod.	QUIC Client: Modified QUIC Server: Modified
P1	✓	✓	✓
P2	✓	✓	✓
P3		✓	✓
O4	✓*		✓

Table 3: Incremental deployability of RAVEN. With certain modifications to the QUIC protocol, RAVEN can offer all three privacy properties at low performance overhead.

3.5 Incremental deployability of RAVEN

We summarize the deployment scenarios in Table 3. RAVEN requires no modification to QUIC or client/server software to achieve lightweight sender anonymity (**P1**) and flow unlinkability (**P2**). Client IPs are encrypted at line rate based on CIDs, and will be different in each connection. Millisecond-level high-frequency key rotation is not required for **P1** and **P2** so the performance overhead is low.

To achieve **P3**, RAVEN requires the participating clients to use a modified QUIC to actively perform CID rotation. The server may remain unmodified, but frequent IP rotation may degrade the performance. This is because (1) QUIC has a rate limit on migration and (2) QUIC performs path validation and resets the congestion window for each migration. To address (1), a minor QUIC configuration change at the server can relax the rate limit. To address (2), we propose a minor change to the QUIC server-side logic (discussed in § 4.1). We argue that the changes we propose for addressing (1) and (2) can benefit other applications with high mobility, e.g., 5G-based vehicular networks. We envision that the changes can be incorporated into the standard QUIC as an extension, which can be enabled at the client's will.

The network can use control groups to differentiate between opt-in and opt-out RAVEN clients. For clients who do not want to use RAVEN, RAVEN can route their traffic normally without doing IP encryption. For clients who want to only use unmodified QUIC, RAVEN can perform IP encryption with low-frequency key rotation to provide **P1** and **P2** with minimized performance overhead.

Since RAVEN performs IP encryption/decryption at the border, the middleboxes managed by the RAVEN-deploying network can still work normally because they are placed inside the network and only see the original (stationary) client IPs and connections. The middleboxes outside the network (on the routing paths, managed by other parties) may fail to track a connection due to IP migration. However, normal QUIC migrations can cause the same issue, because current middleboxes are primarily designed for handling TCP-based protocols, without taking mobility into consideration.

4 RAVEN PROTOTYPE IMPLEMENTATION

Our prototype implementation of RAVEN consists of two parts: a modified QUIC protocol that performs CID rotation, and a data-plane program that performs IP encryption.

4.1 QUIC*: RAVEN-compatible QUIC

Standard IETF QUIC and known QUIC variants do not work effectively with RAVEN. First, connection IDs are persistent per connection in existing QUIC implementations, which makes it trivial for the adversary to link packets to connections. Second, connection migration could incur high overhead: upon detecting a connection migration, the QUIC server resets the congestion window, retransmits all the "in-flight" packets based on the last ACKed packet, and performs path validation [35]. In RAVEN, however, connection migration does not indicate a change in the network path or conditions, making such a mechanism unnecessary. Therefore, we extend the QUIC protocol, which we call QUIC*. QUIC* remains fully compatible with standard QUIC while having extra features: (i) fast active connection ID rotation within a single connection and (ii) maintaining the performance when the connection ID changes.

Actively rotating connection IDs. QUIC* is based on Google's production-ready QUIC implementation QUICHE, which is being used by Chromium and Google's servers [24]. QUIC* only adds around 320 lines of code on top of QUICHE. For better unlinkability, QUIC* provides built-in support for configurable, flexible connection ID rotation. In QUIC*, the QUIC server will send n connection IDs to the client after handshakes, and the QUIC client changes its in-use connection ID to an ID chosen randomly from the n IDs every k (received or sent) packets. The rotation can also be time-based. After p migrations or exhausting all CIDs, the QUIC* client requests a new set of n IDs from the server. QUIC* also has an interface to allow the client application to customize the migration parameters $\langle n, k, p\rangle$. Note that ID rotation only begins after a complete handshake; otherwise, the connection cannot be established successfully.

Identifying migrations caused by RAVEN CID rotation. The migrations caused by RAVEN CID rotation should not trigger the conventional mechanism (i.e., reset congestion window, retransmit packets in-flight, and path validation). To make RAVEN migrations distinguishable, we add a special migration event RAVEN_MIG in QUIC*. In our current implementation, the server classifies a migration as RAVEN_MIG if the client is using IPv6 and its source address changes within the same /64 prefix. A more general approach is to add a special flag in the control field of the client IPv6 address or packet header to notify the server about the migration type. We are in the process of implementing this and leave the complete implementation as future work. We demonstrate how this extension significantly improves performance in § 7.

Discussion. It could be challenging for ordinary clients to choose a good rotation policy to balance the performance-privacy trade-offs. A potential solution is to let the network provide configuration policies (and the corresponding performance overhead) for its clients. QUIC* could expose an interface to allow individual clients to set the policies chosen amongst those preconfigured options. To create efficient policies, the network may profile client traffic based on client activities (website accesses, bulk downloads, etc.) to estimate the trade-offs under different usage scenarios. We leave policy generation as future work.

4.2 Data-plane implementation

RAVEN consists of a software controller for key distribution and rotation, and a $P4_{16}$ data-plane program for encryption.

Line-rate encryption. For our /64 network, we use a 32-bit client ID and a 16-bit padding, as in Figure 2. With the additional 16-bit port number, the encryption input is 64 bits in total. RAVEN currently implements the 2EM encryption scheme developed in PINOT [64]; with more advanced switches, RAVEN could also run 64-bit SIMON/SPECK [5].

The controller, which can run on a dedicated host, generates the encryption keys using the Python urandom function. It uses grpc to communicate with the data plane to update the key table and the key version number in the forwarding table. The keys are stored in a *key* table, the index of which is $\langle$version number, client group index$\rangle$. The version number is stored along with the port forwarding (i.e., switch ingress port to egress port) information in a *forwarding* table so the switch can determine the current version of keys it should use upon receiving a packet. We show the encryption pipeline and the major lookup tables in the P4 program in Figure 3.

Resource usage. Our implementation on an Intel Tofino switch uses all ingress stages and 8% of egress stages. It uses, on average, 48.6% of hash distribution units, 34.9% of logical table IDs, 4.6% of SRAM, and 0% of TCAM. The memory required for maintaining encryption keys is negligible. See § A.5 for more details.

5 RAVEN DEPLOYMENT

We deployed RAVEN in our testbed and evaluated it against the wider Internet. Figure 4 shows our deployment.

QUIC* clients, a RAVEN switch, and a public IPv6 space. The end-host IPv4 client is a Linux server with two AMD EPYC 7302P 3.3Ghz 16-core CPUs and 128GB of memory. We use a QUIC* client and a Chromium browser integrated with QUIC* to run our evaluation. The RAVEN switch is a Wedge 100BF-32X switch with a Tofino programmable chip [45], and it sits between our end-host client and the trusted network's border gateway. The campus border gateway allocates a /64 IPv6 subnet to our network; all RAVEN's encrypted IPv6 addresses are from this address block.

QUIC* server and proxy. We deploy and use our own QUIC* server hosted on an AWS EC2 c4.xlarge instance. We use this server to run microbenchmarks to compare the throughput performance with and without RAVEN (§ 7). We use the same instance as a QUIC* proxy as well to evaluate how well RAVEN defends against webpage fingerprinting attacks when clients visit real websites (§ 6).

Feasibility validation. We set the QUIC* client to migrate every 10 packets and perform key rotation every 500 ms on the RAVEN switch. We used the QUIC* client to download 200 MB and 1 GB files hosted on the QUIC* server 10 times. All files were downloaded successfully through RAVEN, and the SHA1 hash of every file matched that of equivalent downloads directly via unmodified QUIC without RAVEN. We used tcpdump to capture the QUIC* traffic on the server. When fetching the 200 MB file through RAVEN, the server had seen 2,475 connections on average, and all the connections

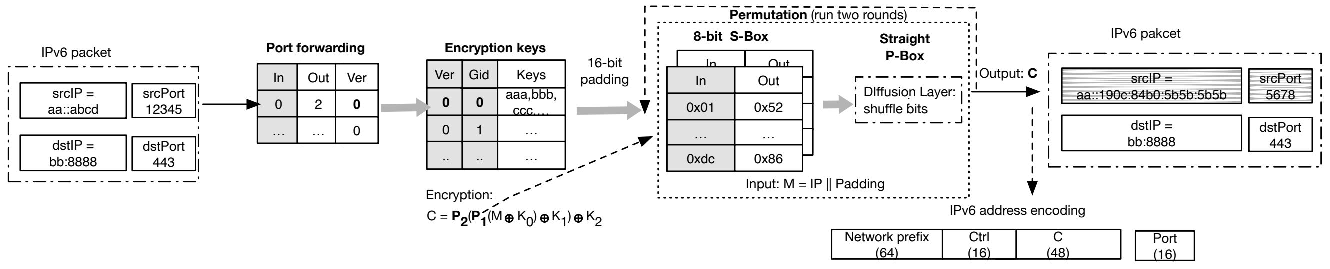

Figure 3: IPv6 source address encryption in RAVEN. Keys of lookup tables are highlighted.

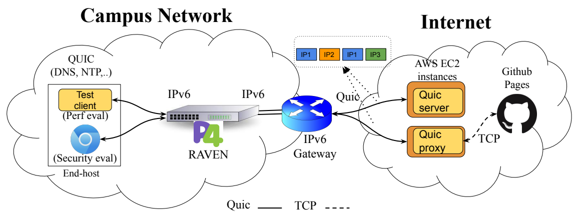

Figure 4: RAVEN deployment setup.

were from unique client IPs. On average each IP was associated with 80 packets (including return packets). When we fixed the key and **only rotated CIDs**, the server had seen 1,525 unique client IPs/connections, with the average subflow length being 130 packets.

Our test suggests RAVEN can effectively obfuscate client IPs, which can prevent IP-based activity and location tracking (**P1** and **P2**). In our evaluation, we focus on more sophisticated attacks that exploit non-IP metadata of flows, i.e, webpage fingerprinting attacks.

6 SECURITY EVALUATION OF RAVEN

Our evaluation focuses on demonstrating the benefits of subflow unlinkability (**P3**) provided by RAVEN. We first describe the scenario, trace collection, and reconstruction attack simulation (§ 6.1 and § 6.2). Then we evaluate the efficiency of state-of-the-art attacks under both real and simulated RAVEN defenses (§ 6.3). We further examine the size of anonymity set using real-world campus traces and discuss where RAVEN stands if an advanced adversary tries to reconstruct a connection from independent sub-connections (§ 6.4 and § 6.5).

6.1 Scenario and experiment setup

Webpage (instead of website) fingerprinting. RAVEN does not hide the destination IPs the clients talk to, but it is still useful for mitigating webpage fingerprinting (WF) attacks.[2] In this setting, the adversary tries to fingerprint the specific webpages the clients are visiting, which share the same IP address (e.g., IPs of CDNs and shared hosting providers) so the adversary cannot infer webpages purely based on their destination IPs. We also assume *SNIs are encrypted* [51]. Importantly, we assume *many clients (connections) are talking to the same server IP simultaneously* (see § 6.4 for anonymity set analysis). In our evaluation, we consider the case that adversary tries to fingerprint GitHub Pages webpages.

The open-world setting. We focus on the open-world setting in our evaluation because it is more realistic. In the open-world setting, the client may visit any webpages *monitored* by the adversary as well as webpages that the adversary has not seen (*unmonitored*) [47]. The adversary aims to infer whether a given trace (statistics extracted from a flow) corresponds to any webpage in the monitored webpages. The open-world setting can be treated as a binary (whether a webpage is in the monitored set of pages) or a multi-class (whether a webpage matches a monitored page) classification problem.

Webpages to fingerprint. To find webpages on GitHub Pages, we search the Common Crawl database [59] (Jan 2022 release) to extract URLs whose domain names end with "*.github.io". From about 0.8 M URLs, we sampled 100 URLs as monitored webpages and 10,000 URLs as unmonitored. We confirmed all webpages are active at the time of our experiment by using curl to check their HTTP status codes; if a webpage's status code was not 200, we replaced it with another webpage sampled from the unused URLs. We visited each unmonitored webpage once and each monitored webpage 20 times, resulting in 12,000 traces total.

A QUIC* proxy and trace collection. To accurately evaluate the privacy benefits of RAVEN, we need to deploy RAVEN and then collect traffic traces as we visit QUIC*-based webpages. However, GitHub does not support QUIC. Instead, we visit a webpage through a reverse proxy that supports QUIC*; the proxy accepts the requests received via QUIC* from the client and fetches resources via TCP, and sends responses back via QUIC*. We compiled QUIC* into the Chromium browser, and set up a QUIC* proxy on an EC2 instance [22, 23]. We launch Chromium via a script to visit webpages through our proxy, and stay for 30 seconds during a visit to ensure the webpage is fully loaded. We used tcpdump to capture the QUIC traffic during a visit on the proxy. The browser cache was cleaned between each visit.

Note on connection ID rotation. We want to use Chromium to collect realistic traffic (as opposed to our customized client software). However, QUIC migration support in Chromium is still experimental at this time, and we can only do timing-based migration

[2]RAVEN can be used with proxies or VPNs to hide destination IPs for better privacy.

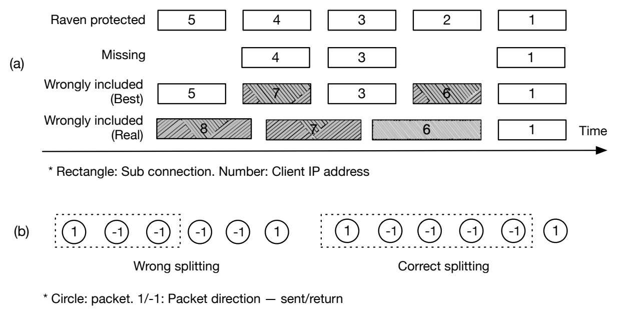

Figure 5: (a) Two reconstruction error types. (b) The correct way to split a 5-packet trace when rotating every 3 packets.

in our test due to an issue in Chromium. [3] In our evaluation, we perform IP migration per 500 ms to ensure good throughput (see § 7 for more details on performance). The average subflow length is about 100 packets.

6.2 Attack simulation

Connection reconstruction attacks. RAVEN does not perform extra obfuscation on the QUIC traffic, so the adversary can distinguish handshake packets from data packets. By monitoring handshake packets, the adversary can identify the first few packets of a connection and the *initial* IP address. This does leak a small amount of information about the connection (also the number of active connections in the network) and could facilitate WF attacks [37, 68]. The adversary then may try to associate observed sub-connections of non-initial IPs to a target initial IP, i.e., performing *reconstruction attacks*, to recover the original connection before further performing traffic analysis.

Given a target connection or initial IP, we define a true positive (TP) as the case that a sub-connection is correctly associated with the target connection (i.e., relevant), otherwise a false negative (FN). Similarly, a true negative (TN) is that a non-relevant sub-connection is correctly identified as non-relevant (otherwise a false positive or FP).

We also refer to the false-negative rate of the attack as *reconstruction error*, denoted by e. We assume a *strong* adversary that is able to perform reconstruction attacks only making the following two errors[4], and defer the possible reconstruction methods to § 6.5.

- *Missing* (M-type): The attack fails to include a few relevant sub-connections in the reconstructed connection, i.e., the attack has only FNs and no FPs.
- *Wrongly-included* (W-type): The attack wrongly replaces some relevant sub-connections with non-relevant sub-connections in the reconstructed connection. This corresponds to an attack that produces the same number of FPs and FNs, and FPs happen to perfectly fill the vacancy of FNs in the reconstructed connection.

We stress that by assuming near-perfect reconstruction attacks, our evaluation of RAVEN's defense is **conservative**. In real-world attacks, a W-type error can cause an avalanche effect. As in Figure 5 (a), if the ending time of sub-connection 6 (non-relevant) is after the beginning of sub-connection 3, the adversary will miss sub-connection 3. This is because sub-connections of the same connection must appear sequentially and cannot overlap each other in timing.

Reconstruction error simulation. We now discuss how to generate the traces produced by reconstruction attacks. Given a sequence of sub-connections and an error rate e, we mark each sub-connection (except for the first one) as M-type or W-type error with a probability of e. If it is a M-type error, we simply remove the sub-connection. For W-type, we replace the sub-connection with a simulated sub-connection, in which the packet size, packet interval, and direction are drawn from the distributions of the real-world traces. The detailed sub-connection simulation method is in Appendix § A.2. Finally, we concatenate the processed sub-connections as the reconstructed connection.

Attack strategy. A non-adaptive adversary may use **(S1)** the first sub-connections or **(S2)** the undefended connections to generate the traces for training. (The adversary can use CIDs to locate her own connections in RAVEN-protected traffic.) An adaptive adversary may take reconstruction errors into consideration, perform reconstruction attacks on her own traffic as if she does not have the ground truth, and use **(S3)** the connections reconstructed under an error rate e for training. The intuition behind S3 is by using reconstructed connections for training, the adversary may minimize the difference between the training and testing traces. Note that for testing, the adversary can only access the first sub-connections or reconstructed connections.

6.3 Efficiency of RAVEN against WF attacks

We first evaluate the efficiency of RAVEN against WF attacks using the real-world traffic produced by RAVEN§ 6.3.1. Then, we simulate RAVEN defense to study how different migration policies affect RAVEN efficiency.

6.3.1 *Evaluation with real-world RAVEN traffic.*

ML-based WF attacks. To better evaluate the efficiency of RAVEN against WF attacks, we consider five attacks that use state-of-the-art machine learning techniques in the evaluation. Deep fingerprinting (DF) attacks [57], timing attacks [50], and Tiktok attacks [50] all use deep learning models. They use packet direction, packet timing, and a combination of direction and packet timing, as features respectively. K-fingerprinting (k-FP) [28] and CUMUL [47] use random forests and SVM with manually-crafted features, respectively.

Evaluation metrics. We primarily use precision-recall (PR) curves and the area under them (AUC-PR) to evaluate the attacks. For k-FP, we use the same method in [57] to define the prediction probability of an example, which is the fraction of the nearest neighbors belonging to the example among the k nearest neighbors. All tests are conducted in the open-world binary-classification setting. Deep-learning-based attacks use 100 epochs and 5,000 as the trace size. Hyperparameter tuning (Appendix Figure 10) shows this configuration balances attack efficiency and training performance.

[3]This issue may involve other components or mechanisms in Chromium and we have been actively working with developers from the QUICHE team to resolve this issue.
[4]Another case is the attack produces no FNs and only FPs. We discuss this case in Appendix § A.3 and leave evaluating it as future work.

	DF	Tiktok	Timing	k-FP
Undef	0.93 (0.01)	0.94 (0.01)	0.96 (0.01)	0.95 (0.01)
S1	0.67 (0.02)	0.75 (0.02)	0.77 (0.04)	0.78 (0.01)
S2 / M	0.55 (0.02)	0.54 (0.03)	0.56 (0.02)	0.46 (0.02)
S2 / W	0.61 (0.02)	0.60 (0.04)	0.63 (0.04)	0.64 (0.02)
S3 / M	0.78 (0.02)	0.81 (0.01)	**0.84** (0.01)	0.82 (0.01)
S3 / W	0.70 (0.02)	0.78 (0.01)	0.84 (0.02)	0.77 (0.02)

Table 4: Average AUC-PR and standard deviation of WF attacks under different attack strategies. The reconstruction error is 0.1. We assume the adversary only makes 1 type of error.

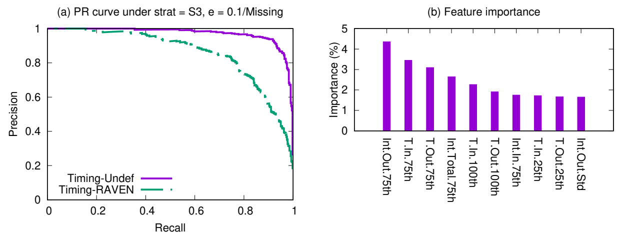

Figure 6: (a) PR curves of the best-performing attack with-/without RAVEN. The PR curves of all attacks under different settings are in Figure 11 in the Appendix. (b) Top 10 features and their importance of the random forests models in k-FP.

We start with fixing the reconstruction error rate to 0.1, and vary error types and attacks. We run the test 10 times and examine the average AUC-PR in each setting.

Baseline performance. We replace the client IPs in a connection with the same IP to simulate the undefended trace. For undefended traces, The AUC-PR of all DL-based attacks and k-FP range from 0.93 to 0.96, as shown in Table 4. CUMUL has the worst performance (in fact, under all parameter combinations). Even for undefended traces, the best AUC-PR is only 0.36. A possible reason is CUMUL relies on packet size features. QUIC has built-in support for packet padding to mitigate traffic analysis. We observe that packet size may leak less information about a connection than other features when using QUICHE, as in our traces the size of 72% of the QUIC packets is 1,238 bytes. If all QUIC packets are well padded to a constant size by the application, the performance of CUMUL could become even worse. In the upcoming sections, we only discuss the results from DL-based attacks and k-FP.

Subflow unlinkability lowers the efficiency of WF attacks. As shown in Table 4, RAVEN successfully reduces the AUC-PRs of all attacks under various attack strategies, especially for non-adaptive attacks. In the best-case scenario, if the adversary adopts the Timing attacks [50], uses reconstructed connections for training (S3), and only makes M-type errors with an error rate of 0.1, the adversary can achieve a relatively high AUC-PR which is 0.84. However, looking at its PR curve, the precision will be less than 0.6 in order to achieve a recall that is greater than 0.9 (Figure 6 (a)). We show the PR curves of the best-case performance of all attacks under different settings in Figure 11 in the Appendix.

Timing features play a crucial role in the classification. Timing and k-FP are able to achieve better AUC-PRs than other attacks in tests. Examining the feature importance of the random forests model used in the best-performing k-FP attack, we find that timing-related features contribute the most to the classification. As in Figure 6 (b), the top 10 important features are all timing-related. In fact, 16 of the top 20 features are statistics about packet inter-arrival times or timestamps. The importance of the two types of features alone (24 features out of 175 features) is 45%.

Using reconstructed connections for training (S3) overall can make the attacks more efficient. One possible explanation is this strategy may reduce the difference between the amount of information provided in the training and test traces. Compared to S3, using the original connections for training (S2) ignores the information loss caused by traffic splitting, and thus fails to accurately model RAVEN-protected traffic.

	Bin-class	Multi-class	
	AUC-PR	r-Precison	Recall
Undef	0.96 (0.01)	0.88 (0.06)	0.87 (0.05)
Const-1	0.86 (0.01)	0.78 (0.09)	0.74 (0.10)
Const-20	0.85 (0.07)	0.77 (0.08)	0.74 (0.11)
Random	0.78 (0.02)	0.71 (0.09)	0.44 (0.14)
w/ Front	0.58 (0.06)	0.42 (0.12)	0.36 (0.04)
w/ WTF-PAD	0.64 (0.13)	0.57 (0.02)	0.43 (0.06)

Table 5: Efficiency of the best-performing attacks against RAVEN under different migration policies. $e = 0.1$.

Multi-class open-world. We further consider the open-world setting as a multi-class classification problem, i.e., the classifier needs to determine not only whether a given instance is in the monitored set but also the exact site. We used r-precision and recall proposed in [66], with $r = 5$ based on our dataset, as the metrics. Our observation remains unchanged: Timing attacks under S3/M outperform all the other attacks, with an r-precision of 0.76 (STD=0.07), dropped from 0.88 (STD=0.06) against undefended traces. The recall drops more dramatically, from 0.87 (STD=0.05) to 0.56 (STD=0.05), suggesting the attacks will produce a considerable amount of FNs.

Impact of error rate on RAVEN efficiency. We pick the best-performing attack (Timing attacks using S3 with M-type error) and examine how error rate affects attack efficiency (0.1 to 0.9 with a step of 0.1). The AUC-PRs drop from about 0.84 to 0.77 as the error rate increases, which is expected because S1 is the worst-case of S3/M.

6.3.2 Evaluation with simulated RAVEN defense.

RAVEN defense simulation. Recall that due to the aforementioned Chromium issues, our testbed can only adopt limited migration policies. To study the efficiency of RAVEN against WF attacks under diverse, finer-grained migration policies, we need to simulate the RAVEN defense. To improve simulation accuracy, we determine the splitting boundary in the following way: if the migration frequency is every k packets, a sub-connection contains k packets plus all the return packets before the first outgoing packet after the k packets. The reason is *the return packets from the server are always sent to the last-seen client IP*, and the client IP only changes in the

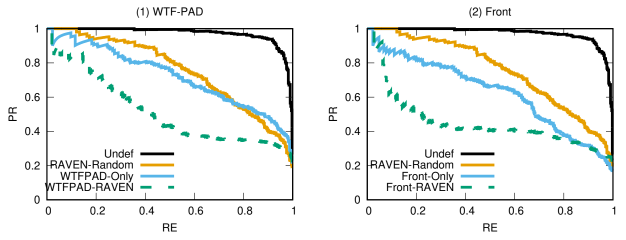

Figure 7: The PR curves of timing attacks against undefended traces, traces protected with WTF-PAD/Front only, and traces protected with both RAVEN and WTF-PAD/Front. We show the PR curves under the best-case defenses. $e = 0.1$.

next outgoing packet even if the rotation threshold has been hit. We give an example in Figure 5(b).

Constant-frequency and random migration. Since the splitting boundary is always at outgoing/sent packets, the finest way to control migration is based on the number of outgoing packets, rather than the total number of packets or timing.[5] In our discussion, our migration policy is based on a counter of outgoing packets. We use k to denote migration frequency.

Random migration is very effective. When the client migrates with k randomly selected from 1 to 20, the AUC-PR is 0.78 (STD=0.02). In the multi-class setting, the r-precision becomes 0.71 (STD=0.09) and the recall is 0.44 (STD=0.14).

For constant-frequency migration, when $e = 0.1$, the AUC-PRs are similar under different k ($k = 1$ to 20). It seems to be counter-intuitive that higher frequency migration does not offer better privacy. We noticed that the result could be an artifact of our reconstruction attack simulation. See Appendix § A.4 for more details. As we demonstrate soon in § 6.5, high-frequency migration can increase the error rate of reconstruction attacks. When e increases from 0.1 to 0.5, the AUC-PR when $k = 1$ falls to 0.65.

Combination of RAVEN and application-layer defenses. RAVEN is complementary to application-layer WF defenses, as RAVEN is an IP and transport layer defense. We can combine RAVEN with other WF defenses to achieve better privacy (assuming that Tor or VPNs have replaced TCP with QUIC as the transport). We consider two efficient defenses: Front [18] and WTF-PAD [38]. Both defenses are lightweight, application-layer defenses and mitigate WF attacks by adding paddings. We do not consider WF defenses that enforce a constant packet rate or add delays between packets that has high performance overheads (e.g., Tamaraw[8]). We used the default parameters in WTF-PAD, as in the code provided by [18]. For Front, we changed W_{min} to 0.5 so more noise will be added to the first sub-connection. The bandwidth overheads of both defenses are about 250%.

Without RAVEN, the average AUC-PRs of the best-performing attacks (S3/W, e=0.1) against WTF-PAD and Front are 0.77 (STD=0.05) and 0.70 (STD=0.04), respectively. The performance of RAVEN using random migration is comparable to WTF-PAD. Even combined with weak RAVEN defense (constant migration per 20 packets), the average AUC-PRs drop to 0.64 (STD=0.13) and 0.58 (STD=0.06). The results suggest RAVEN-protected clients could adopt application-layer defenses to further mitigate WF attacks. The PR curves of the attacks under the best-case defenses are shown in Figure 7.

[5]For example, in Figure 5(b), migrating per 1 to 5 packets produces the same splitting as migration per 1 outgoing packet. Similarly, one can migrate at any time before the 2^{nd} outgoing packet but the first sub-connection always consists of packets 1 to 5.

6.4 Insights on anonymity sets

To understand the anonymity set size in the real world, we captured 300 seconds of anonymized UDP IPv4 traffic sent to port 443 at the beginning of every hour, on a large campus network (no IPv6 support) between Oct 31, 2022 and Nov 13, 2022. We extracted 33.5 M QUIC connections using Wireshark 3.6.5. In this section, we examine whether real-world QUIC traffic can form reasonable anonymity sets for RAVEN to achieve the desired privacy goals.

Anonymity set for P1 and P2. The anonymity sets for **P1** and **P2** are the active clients in the network. If the network only has one active client, it is trivial to link IPs or connections to the client. We examined the number of active clients in any given 1-second time window, which ranges from 4 to 1,519, and calculated the average number across 300 seconds in each hour. The result (shown in Figure 8 (**upper**)) exhibits strong hourly and daily patterns: there are more active clients around noon of a day and on weekdays.

Anonymity set for P3. For **P3**, the anonymity set size is the number of active QUIC connections to the same server IP. If a set of connections of the same server IP have packets sent or received during the same 1-second time period, we consider them as concurrent. (Server port is always 443.) In each hour, we counted the average number of concurrent connections per second per server IP in the 300-second time period. As shown in Figure 8 (**middle**), in any given 1-second time period, a server on average has 1.3 to 3.3 concurrent connections. We observe the same hourly and daily patterns as for the anonymity sets for **P1** and **P2**. Depending on the services and time, the anonymity set changes significantly. For example, Akamai has 5 to 50 connections per server per second while CloudFront has at most 4. A server of google.com can have up to 1,066 concurrent connections. Looking at those most-seen server/destination IPs belonging to CDN and DNS services, at least 20% of the time those IPs have more than two (up to 120) concurrent connections per second, except for AWS CloudFront (Appendix Figure 12).

Takeaways. We estimate that the QUIC traffic currently only accounts for about 20% of all campus traffic, including both TCP and UDP traffic. Even so, the current QUIC traffic provides sufficient large anonymity sets for the privacy proprieties RAVEN aims to achieve (we will soon demonstrate that the anonymity set size is sufficient for **P3** in § 6.5). We believe as the Internet gradually transitions from TCP to QUIC, more clients and popular websites will adopt QUIC, which can provide larger anonymity sets. Our results are based on a single network, but can still shed light on the benefits of RAVEN in similar scale deployments.

6.5 Understanding reconstruction efficiency

As shown in Table 4, without doing reconstruction, a non-adaptive adversary cannot perform efficient WF attacks. In this section, we discuss two possible ways to perform reconstruction.

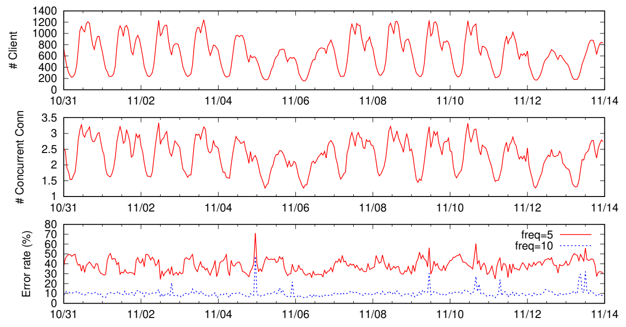

Figure 8: Anonymity set sizes and error rates over 2-week time period (starting from a Monday, measured hourly). The plot shows one x axis tick every 12 hours.

Timing-based reconstruction. In RAVEN-protected traffic, *sub-connections of the same connection must appear sequentially so overlapping sub-connections can never belong to the same connection.* This simple fact can be used by the adversary to eliminate non-relevant sub-connections. Then, the adversary may simply classify a given sub-connection and the first sub-connection seen after it as the same connection, and iterate this process until the reconstructed connection contains a certain number of packets. We simulated this greedy attack using the connections associated with the servers of the most connections in each hour. During the simulation, we give the adversary an advantage over RAVEN by assuming the adversary knows the exact lengths of target connections in advance.

When $k = 5$, the attacks misidentified at least 24.3%–70.9% of the sub-connections, with an average of 38.3%. Reducing the migration frequency decreases the error rate: the error rate ranges from 5.4% to 47.6%, with an average of 10.1% when $k = 10$. The error rate estimation is very conservative, as in practice the adversary may not be able to have accurate prior knowledge on connection length. Imagine a target connection is only 100-packet long while some traffic-analysis attacks require 500 packets for good accuracy, the adversary may incorrectly associate 400 non-relevant packets to the target connection.

An interesting observation is *larger anonymity set does not necessarily result in higher reconstruction error.* Generally, the reconstruction attacks are likely to be less accurate when more non-relevant sub-connections could be mixed with the relevant sub-connections. Even with fewer concurrent connections, the extent of mixing can be high, depending on the timing of packets. Based on this observation, a single active client may create "decoy" connections to confuse the adversary to protect a privacy-sensitive connection. By migrating the decoy connections at strategic points (e.g., between two sub-connections of the sensitive connection), the decoy connections are mixed with the sensitive connection to make reconstruction attacks harder. This can be viewed as a variant of Front or WTF-PAD. Overall, we argue that (1) greedy reconstruction may not be reliable, and (2) higher-frequency migration may make reconstruction attacks less accurate, and consequently can worsen WF attack performance.

Metadata-based reconstruction. We examined known metadata that could link packets to connections, namely QUIC CIDs in the return packets, QUIC packet number, and IP IDs. CIDs in the return packets and QUIC packet numbers are randomized/encrypted by default in QUIC, and QUIC IPv6 traffic does not contain IP IDs. These metadata cannot be used for reconstruction. See detailed discussions in § A.1.

7 PERFORMANCE EVALUATION

Recall that the QUIC* configurable migration parameters are $\langle n, k, p\rangle$: the client uses one connection ID from n negotiated IDs every k packets. After p ID rotations, the client requests a new set of n IDs from the server. We use the wide-area testbed to understand the performance impact of QUIC* parameters and encryption key rotation frequency on client throughput, by using a QUIC* client to fetch a 200 MB file hosted on the remote QUIC* server. Results are shown in Figure 9. By default n is set to 10 in the tests. For a given setting, we repeat the measurement 20 times and report the median throughput. We define $\bar{t} = 1/t$ as the key rotation frequency, or the number of key rotations per second. The connection ID rotation frequency is $\bar{k} = 100/k$, or the number of ID rotations per 100 packets. The ID set update frequency is defined as $\bar{p} = 10/p$ — the number of updates per 10 ID rotations.

Unmodified QUIC and QUIC* under high-frequency key rotation. We first examine how key rotation frequency affects client throughput. For a given $\bar{t}$, the lifetime of an encryption key is $(N_k + 1) * t$ (N_k is the number of historical keys maintained by the switch), and return packets encrypted under this key cannot be decrypted and will be dropped by the switch after $(N_k + 1) * t$. In our test, $N_k = 2$. We can see from Figure 9 (a) (only showing the results under $\bar{t}$ up to 10 to demonstrate the trend), the throughput of unmodified QUIC decreases dramatically as IP migration happens more frequently, and the probability of connection timeout due to too many retransmission timeouts also increases.

For QUIC*, we fix the connection ID to eliminate the potential overhead introduced by ID rotation. As in Figure 9 (a), when $\bar{t} \leq 20$ (or key rotation per 50 ms), the median throughput is similar to that of the control case, which is using unmodified QUIC without RAVEN ($\bar{t} = 0$). But, as $\bar{t}$ becomes higher, we do see more "outlier" cases with degraded throughput. This is because extremely high $\bar{t}$ could make most of the return packets get dropped due to key expiration and trigger retransmission. Overall, by distinguishing IP migration caused by RAVEN from normal migration, QUIC* reduces performance overhead significantly.

The results also shed light on the client throughput with timing-based migration. Recall that in our deployment $\bar{t} = 2$ or IP migration every 500 ms. The throughput degradation of QUIC* is only 0.5%, while that of the unmodified QUIC is 12%. If the client is modified and the server remains unmodified, the performance overhead is also about 12%.

Fine-grained high-frequency connection ID rotation. We fix key rotation frequency $\bar{t} = 0$ and ID set update frequency $\bar{p} = 0$ and vary $\bar{k}$ to examine the overhead introduced by active ID rotation. In Figure 9 (b), when $\bar{k}$ increases from 0.5 to 100, the throughput drops by 20%, from 260 Mbps to 206 Mbps. This unexpected degradation is likely caused by the client: The client needs to select a new ID and update the in-use connection ID every k packets. This routine

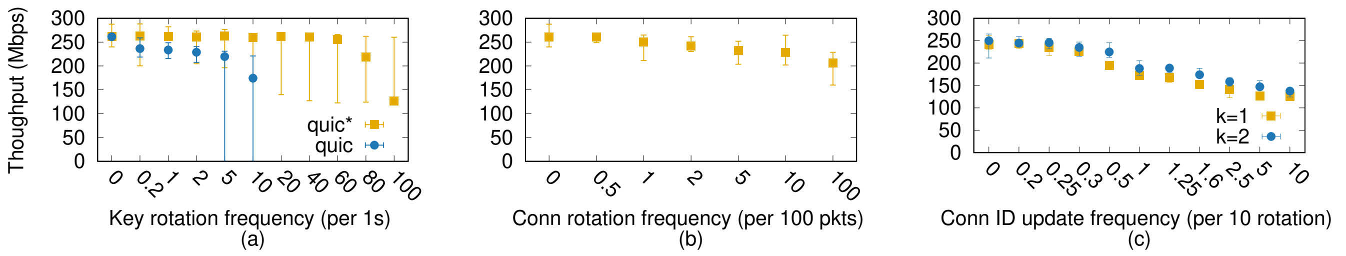

Figure 9: Median client throughput under different controlled settings. Error bars indicate min and max. (a) No connection ID rotation, and vary key rotation frequency (or IP migration frequency) $\bar{t}$. (b) No key rotation and connection ID set update, vary connection ID rotation frequency $\bar{k}$. (c) No key rotation, vary connection ID set update frequency p under $k = 1$ and $k = 2$. In (a) the throughput of unmodified QUIC drops to almost zero when key rotation frequency ≥ 20 so we omit those data points.

happens more frequently as $\bar{k}$ increases. The corresponding code can be optimized to make this routine more efficient.

Connection ID set update frequency. Each connection ID set update requires 1 RTT to complete, so the connection ID update frequency affects the number of extra RTTs, which is the dominant source of performance degradation. As in Figure 9(c), higher ID rotation frequency incurs more ID updates, leading to lower throughput. When using each connection ID only once ($\bar{p} = 1$), the throughput degradation is about 28% for $k = 1$. Overall, the number of additional RTTs is determined by k and p. For every 100 packets, the additional RTTs is $\bar{k} * \bar{p}/10$. To reduce the connection ID set update frequency, QUIC* can prepare a large number of CIDs during connection ID negotiation. For the 8-byte CIDs used in QUICHE, a QUIC frame can hold about 160 CIDs.

8 RELATED WORK

In this section, we discuss related work not covered in § 2.

Network-based solutions: Hiding IP addresses and traffic patterns. Network-layer anonymity systems, such as LAP [32], Dovetail [54], HORNET [9], PHI [11], and TARANET [10], can provide stronger sender/receiver anonymity and flow unlinkability, but they do not provide subflow unlinkability. SPINE [14] uses programmable switches to encrypt IP addresses and Ditto [41] leverages programmable switches to obfuscate traffic. They both require cooperation from the receiving-end networks to decrypt/decode obfuscated packets.

While RAVEN focuses on protecting client IP addresses against privacy attacks, a type of moving target defense randomizes server IP addresses for security purposes (e.g., mitigating DDoS attacks), such as Mirage and NASR [2, 4, 36, 43]. Such systems are not designed to provide subflow unlinkability or even flow unlinkability.

Client-based solutions: Traffic splitting. The split-based solutions, such as splitting a flow to use multiple routing paths [29, 39, 65], are typically used in combination with tunnel-based solutions (e.g., Tor [16]) for mitigating traffic-analysis attacks; alone, they do not provide anonymity. The split-based solutions do achieve subflow unlinkability to some extent. However, a recent study from Beckerle et al. [6] performs a critical evaluation of splitting-based defenses, and demonstrates that the defenses are less effective once the adversary is able to observe the vast majority of traffic. Beckerle et al. [6] develop novel Maturesc attacks against existing splitting-based defenses that can achieve good accuracy. We leave evaluating Maturesc against RAVEN as future work.

MIMIQ and PINOT. MIMIQ [25] is the first system to use QUIC's connection migration feature to change IPv4 addresses to improve privacy. RAVEN goes much further in extending QUIC to achieve packet unlinkability, by changing QUIC's connection ID. PINOT [64] proposes in-network address obfuscation only for connectionless traffic (e.g., DNS), while RAVEN supports connection-oriented traffic. In addition, our paper proposes a novel division of labor between clients and switches, and builds and deploys a prototype system. We also present a full-scale security and performance evaluation with real hardware and traffic.

9 CONCLUSION

RAVEN is a privacy solution for enterprise networks, allowing them to provide privacy-as-a-service to their clients. RAVEN hides client identifiers and breaks the association between packets that belong to the same connection by rapidly rotating the IP address, port number, and connection ID a client uses publicly on the Internet. RAVEN lets the client decide when to change its connection metadata, while the network actually performs the rotation. Our deployment and evaluation show RAVEN can enhance the privacy of the enterprise network and its clients with low overhead.

ACKNOWLEDGMENTS

We are grateful to Noah Apthorpe, Hooman Mohajeri Moghaddam, and Paul Schmitt for insights that helped us refine this work. We are also grateful to anonymous reviewers for their invaluable feedback. This work was supported by the National Science Foundation under grants CNS-1553437 and CNS-1704105, and by DARPA under grant FA8750-19-C-0079.

REFERENCES

[1] S. Amante, B. Carpenter, S. Jiang, and J. Rajahalme. 2011. *IPv6 Flow Label Specification.* RFC 6437. RFC Editor.
[2] Spyros Antonatos, Periklis Akritidis, Evangelos P Markatos, and Kostas G Anagnostakis. 2007. Defending against hitlist worms using network address space randomization. *Computer Networks* 51, 12 (2007), 3471–3490.
[3] Apple. 2021. Apple iCloud Private Relay Overview. https://www.apple.com/privacy/docs/iCloud_Private_Relay_Overview_Dec2021.PDF.
[4] Nahid Bandi, Hesam Tajbakhsh, and Morteza Analoui. 2021. FastMove: Fast IP switching moving target defense to mitigate DDOS attacks. In *IEEE Conference*

on Dependable and Secure Computing (DSC). IEEE, 1–7.

[5] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. 2013. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Paper 2013/404. https://eprint.iacr.org/2013/404 https://eprint.iacr.org/2013/404.

[6] Matthias Beckerle, Jonathan Magnusson, and Tobias Pulls. 2022. Splitting Hairs and Network Traces: Improved Attacks Against Traffic Splitting as a Website Fingerprinting Defense. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society*. 15–27.

[7] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, François-Xavier Standaert, John Steinberger, and Elmar Tischhauser. 2012. Key-alternating ciphers in a provable setting: Encryption using a small number of public permutations. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 45–62.

[8] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014. A systematic approach to developing and evaluating website fingerprinting defenses. In *ACM SIGSAC Conference on Computer and Communications Security*. 227–238.

[9] Chen Chen, Daniele E Asoni, David Barrera, George Danezis, and Adrain Perrig. 2015. HORNET: High-speed onion routing at the network layer. In *ACM SIGSAC Conference on Computer and Communications Security*. 1441–1454.

[10] Chen Chen, Daniele E Asoni, Adrian Perrig, David Barrera, George Danezis, and Carmela Troncoso. 2018. TARANET: Traffic-analysis resistant anonymity at the network layer. In *IEEE European Symposium on Security and Privacy*. IEEE, 137–152.

[11] Chen Chen and Adrian Perrig. 2017. PHI: Path-Hidden Lightweight Anonymity Protocol at Network Layer. In *Privacy Enhancing Technologies*. 100–117.

[12] Kenjiro Cho, Koushirou Mitsuya, and Akira Kato. 2000. Traffic Data Repository at the WIDE Project. In *USENIX Annual Technical Conference*.

[13] Alissa Cooper, Fernando Gont, and Dave Thaler. 2016. Security and privacy considerations for IPv6 address generation mechanisms. *IETF, RFC 7721* (2016).

[14] Trisha Datta, Nick Feamster, Jennifer Rexford, and Liang Wang. 2019. SPINE: Surveillance Protection in the Network Elements. In *USENIX Workshop on Free and Open Communications on the Internet*.

[15] S. Deering and R. Hinden. 2017. *Internet Protocol, Version 6 (IPv6) Specification*. STD 86. RFC Editor.

[16] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. *Tor: The second-generation onion router.* Technical Report. Naval Research Lab Washington DC.

[17] DNSCrypt. 2020. DNSCrypt. https://dnscrypt.info/.

[18] Jiajun Gong and Tao Wang. 2020. Zero-delay Lightweight Defenses against Website Fingerprinting. In *USENIX Security Symposium*. 717–734.

[19] F. Gont. 2014. *A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)*. RFC 7217. RFC Editor.

[20] F. Gont and T. Chown. 2016. *Network Reconnaissance in IPv6 Networks*. RFC 7707. RFC Editor.

[21] F. Gont, S. Krishnan, T. Narten, and R. Draves. 2021. *Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6*. RFC 8981. RFC Editor.

[22] Google. 2022. The Chromium Projects. https://www.chromium.org/chromium-projects/.

[23] Google. 2022. Playing with QUIC. https://www.chromium.org/quic/playing-with-quic/.

[24] Google. 2022. QUICHE. https://quiche.googlesource.com/quiche/.

[25] Yashodhar Govil, Liang Wang, and Jennifer Rexford. 2020. MIMIQ: Masking IP with Migration in QUIC. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*.

[26] Benjamin Greschbach, Tobias Pulls, Laura M Roberts, Philipp Winter, and Nick Feamster. 2017. The Effect of DNS on Tor's Anonymity. In *Network and Distributed System Security Symposium*. Internet society.

[27] Stephen Groat, Matthew Dunlop, Randy Marchany, and Joseph Tront. 2010. The privacy implications of stateless IPv6 addressing. In *Annual Workshop on Cyber Security and Information Intelligence Research*. 1–4.

[28] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *USENIX Security Symposium*. USENIX Association, Austin, TX, 1187–1203.

[29] Sébastien Henri, Gines Garcia-Aviles, Pablo Serrano, Albert Banchs, and Patrick Thiran. 2020. Protecting against Website Fingerprinting with Multihoming. In *Privacy Enhancing Technologies*, Vol. 2020. 89–110. https://doi.org/10.2478/popets-2020-0019

[30] Dominik Herrmann, Christian Banse, and Hannes Federrath. 2013. Behavior-based tracking: Exploiting characteristic patterns in DNS traffic. *Computers & Security* 39 (2013), 17–33.

[31] Andrew Hintz. 2002. Fingerprinting websites using traffic analysis. In *International Workshop on Privacy Enhancing Technologies*. Springer, 171–178.

[32] Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim, Adrian Perrig, Akira Yamada, Samuel C Nelson, Marco Gruteser, and Wei Meng. 2012. LAP: Lightweight anonymity and privacy. In *IEEE Symposium on Security and Privacy*. IEEE, 506–520.

[33] C. Huitema, S. Dickinson, and A. Mankin. 2022. *DNS over Dedicated QUIC Connections*. RFC 9250. RFC Editor.

[34] Intel Tofino 2021. Intel Tofino Chip. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series/tofino.html.

[35] J. Iyengar and M. Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. https://www.rfc-editor.org/rfc/rfc9000.html. RFC 9000.

[36] Quan Jia, Kun Sun, and Angelos Stavrou. 2013. Motag: Moving target defense against internet denial of service attacks. In *International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–9.

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

[38] Marc Juárez, Mohsen Imani, Mike Perry, Claudia Díaz, and Matthew Wright. 2016. Toward an Efficient Website Fingerprinting Defense. In *European Symposium on Research in Computer Security (Lecture Notes in Computer Science, Vol. 9878)*. Springer, 27–46. https://doi.org/10.1007/978-3-319-45744-4_2

[39] Wladimir De la Cadena, Asya Mitseva, Jens Hiller, Jan Pennekamp, Sebastian Reuter, Julian Filter, Thomas Engel, Klaus Wehrle, and Andriy Panchenko. 2020. TrafficSliver: Fighting Website Fingerprinting Attacks with Traffic Splitting. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[40] Shuai Li, Huajun Guo, and Nicholas Hopper. 2018. Measuring information leakage in website fingerprinting attacks and defenses. In *ACM SIGSAC Conference on Computer and Communications Security*. 1977–1992.

[41] Roland Meier, Vincent Lenders, and Laurent Vanbever. 2022. ditto: WAN Traffic Obfuscation at Line Rate. In *NDSS Symposium 2022*. Internet Society.

[42] Vikas Mishra, Pierre Laperdrix, Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Martin Lopatka. 2020. Don't count me out: On the relevance of IP address in the tracking ecosystem. In *World Wide Web Conference*. 808–815.

[43] Prateek Mittal, Dongho Kim, Yih-Chun Hu, and Matthew Caesar. 2011. Mirage: Towards deployable DDoS defense for Web applications. *arXiv preprint arXiv:1110.1060* (2011).

[44] T. Narten, R. Draves, and S. Krishnan. 2007. *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. RFC 4941. RFC Editor.

[45] Edgecore Networks. 2019. Edge-core Wedge 100BF-32X. https://www.edge-core.com/productsInfo.php?cls=1&cls2=5&cls3=181&id=335,2019.

[46] University of Oregon. 2020. Route Views Archive Project. http://archive.routeviews.org/.

[47] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website Fingerprinting at Internet Scale.. In *NDSS*.

[48] Simran Patil and Nikita Borisov. 2019. What can you learn from an IP?. In *Applied Networking Research Workshop*. 45–51.

[49] Barath Raghavan, Tadayoshi Kohno, Alex C. Snoeren, and David Wetherall. 2009. Enlisting ISPs to Improve Online Privacy: IP Address Mixing by Default. In *Privacy Enhancing Technologies Symposium*, Ian Goldberg and Mikhail J. Atallah (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 143–163.

[50] Mohammad Saidur Rahman, Payap Sirinam, Nate Mathews, Kantha Girish Gangadhara, and Matthew Wright. 2020. The Utility of Packet Timing in Website Fingerprinting Attacks. *Privacy Enhancing Technologies* 2020, 3 (2020), 5–24. https://doi.org/doi:10.2478/popets-2020-0043

[51] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. 2022. TLS Encrypted Client Hello. https://datatracker.ietf.org/doc/draft-ietf-tls-esni/.

[52] RIPE NCC. 2017. Best Current Operational Practice for Operators: IPv6 prefix assignment for end-users — persistent vs non-persistent, and what size to choose. https://www.ripe.net/publications/docs/ripe-690.

[53] Erik Rye, Robert Beverly, and KC Claffy. 2021. Follow the scent: Defeating IPv6 prefix rotation privacy. In *ACM Internet Measurement Conference*. 739–752.

[54] Jody Sankey and Matthew Wright. 2014. Dovetail: Stronger anonymity in next-generation internet routing. In *Privacy Enhancing Technologies Symposium*. Springer, 283–303.

[55] James Saxon and Nick Feamster. 2022. GPS-based Geolocation of consumer IP addresses. In *International Conference on Passive and Active Network Measurement*. Springer, 122–151.

[56] Shodan. 2020. The search engine for the Internet of Things. https://www.shodan.io/.

[57] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada). 1928–1943. https://doi.org/10.1145/3243734.3243768

[58] Alex C Snoeren and Hari Balakrishnan. 2000. An end-to-end approach to host mobility. In *International Conference on Mobile Computing and Networking*. 155–166.

[59] The Common Crawl team. 2022. The Common Crawl Dataset. https://commoncrawl.org/.

[60] tele2. 2020. Tele2 Speedtest. http://speedtest.tele2.net/.

[61] M. Thomson and S. Turner. 2021. *Using TLS to Secure QUIC*. RFC 9001. RFC Editor.

[62] S. Thomson, T. Narten, and T. Jinmei. 2007. *IPv6 Stateless Address Autoconfiguration*. RFC 4862. RFC Editor. http://www.rfc-editor.org/rfc/rfc4862.txt http://www.rfc-editor.org/rfc/rfc4862.txt.

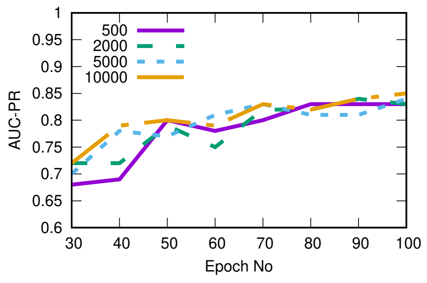

Figure 10: Average PR-AUCs of timing attacks under varied epoch number and trace size. In our tests the epoch number is 100 and trace size is 5,000.

[63] Liang Wang. 2023. RAVEN Github Repo. https://github.com/liangw89/RAVEN.

[64] Liang Wang, Hyojoon Kim, Prateek Mittal, and Jennifer Rexford. 2021. Programmable in-network obfuscation of DNS traffic. In *NDSS: DNS Privacy Workshop.*

[65] Mona Wang, Anunay Kulshrestha, Liang Wang, and Prateek Mittal. 2022. Leveraging strategic connection migration-powered traffic splitting for privacy. *Proceedings on Privacy Enhancing Technologies* 3 (2022), 498–515.

[66] Tao Wang. 2020. High precision open-world website fingerprinting. In *IEEE Symposium on Security and Privacy (SP).* IEEE, 152–167.

[67] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *USENIX Security Symposium.* San Diego, CA, 143–157.

[68] Tao Wang and Ian Goldberg. 2016. On Realistically Attacking Tor with Website Fingerprinting. *Proc. Priv. Enhancing Technol.* 2016, 4 (2016), 21–36.

[69] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. 2005. Profiling internet backbone traffic: Behavior models and applications. *ACM SIGCOMM Computer Communication Review* 35, 4 (2005), 169–180.

[70] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. 2008. Internet traffic behavior profiling for network security monitoring. *IEEE/ACM Transactions On Networking* 16, 6 (2008), 1241–1252.

[71] Ting-Fang Yen, Yinglian Xie, Fang Yu, Roger Peng Yu, and Martin Abadi. 2012. Host Fingerprinting and Tracking on the Web: Privacy and Security Implications.. In *NDSS*, Vol. 62. 66.

[72] Fangming Zhao, Yoshiaki Hori, and Kouichi Sakurai. 2007. Analysis of privacy disclosure in DNS query. In *International Conference on Multimedia and Ubiquitous Engineering.* IEEE, 952–957.

[73] Sebastian Zimmeck, Jie S Li, Hyungtae Kim, Steven M Bellovin, and Tony Jebara. 2017. A privacy analysis of cross-device tracking. In *USENIX Security Symposium.* 1391–1408.

A APPENDIX

A.1 Possible identifiers in QUIC

IPv6 flow label and extension headers. To better understand the use of flow labels and extension headers, we examined three 15-min MAWI anonymized network traces [12], collected in August 2021. Out of around 320 K IPv6 TCP flows, 87% of them have zero-value flow labels. Meanwhile, only 99 IPv6 packets contain extension headers. Other than zeroing, we can also extend RAVEN to obfuscate the flow label field, in a similar fashion we do port obfuscation, and randomize the order of extension headers.

QUIC packet number. A QUIC packet has a packet number that increases per packet transmitted, which is used for tracking packets and locating the cryptographic nonce for packet protection. Even with IP and connection ID rotation, the continuity of packet numbers of two sub-connections may reveal to the adversary that they belong to the same connection. However, QUIC performs header protection by default [61], so the packet numbers are encrypted and cannot be learned by the adversary.

IP identification number. Similar to QUIC packet number, the monotonically increasing IP identification (IP ID) field of the IPv4 header can also be used for associating packets to connections. In IPv6, regular IPv6 packets do not contain IP IDs, while the fragmented IPv6 packets have an identification field that is the same as IP ID. However, QUIC RFC requires "UDP datagrams MUST NOT be fragmented at the IP layer." (see [35], Sec 14). Different from IPv4, only endpoints can perform IP fragmentation in IPv6. QUIC endpoints will adjust the packet size to obey the MTUs of their networks to avoid fragmentation. When detecting the MTU of a network is smaller than QUIC's maximum datagram size, QUIC will close the connection. In conclusion, QUIC IPv6 traffic does not contain IP ID.

Connection IDs in return packets. In QUIC, each peer of a connection independently selects the connection IDs that its counterpart should use, and then they exchange the set of connection IDs during their handshake phase. Thus, theoretically, RAVEN should also rotate the connection IDs in the packets sent from the server because they can also be used to identify clients and connections. However, the connection IDs in server-to-client packets are not required and do not need to be set by the server. In fact, in Google's QUICHE implementation, server-to-client packets do not even contain connection IDs.

A.2 W-type error simulation

We obtain five distributions from the real-world traces: packet direction, sent packet size, return packet size, packet interval of the packets of the same direction, and packet interval of the packets of different directions. During simulation, we first draw the direction of a packet from the direction distribution, and then draw the packet size from the corresponding packet size distribution based on the packet direction. Then, we compare the direction of the current packet with its preceding packet to determine the interval distribution to draw from. The timestamp of the packet is the timestamp of its preceding packet plus the selected time interval. We keep creating new packets until the timestamp of the packet is greater than that of the first packet of the succeeding sub-connection.)

A.3 Discussions on attack evaluation

Reconstruction attacks with FPs only. In this case, non-relevant sub-connections are inserted between relevant sub-connections in the reconstructed connection. We leave the evaluation of this as future work. We suspect this error may make further WF attacks less efficient than M-/W-type errors, because the connections reconstructed under this error are similar to the connections protected by a weak version of application-layer defenses(e.g., Front and WTF-PAD [18, 38]).

Other migration strategies. We stress that our evaluation is **conservative**, because we model strong reconstruction attacks. In practice, reconstruction attacks could become far less accurate and therefore make WF attacks less effective.

One migration strategy we didn't discuss in the paper is intentionally delaying the new sub-connections so that the sub-connections appear to be more independent. A drawback of such a strategy is extra latency. One may migrate strategically to minimize latency

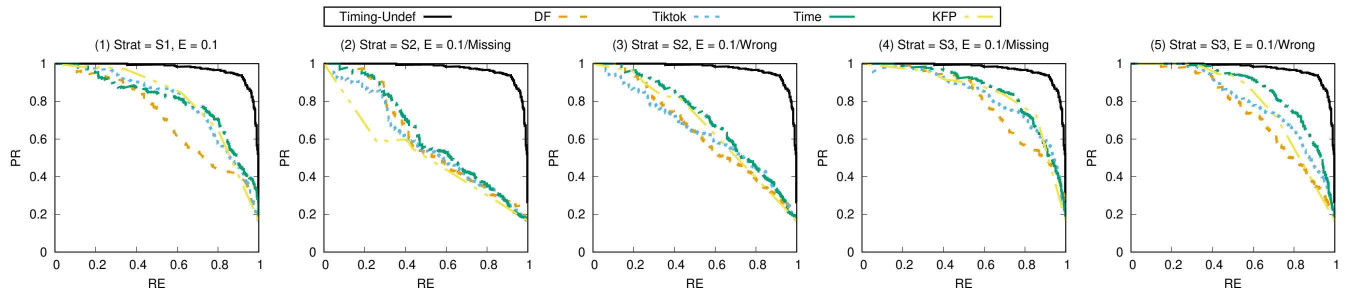

Figure 11: The precision-recall curves of WF attacks against RAVEN under different strategies. The plots are based on the best-case performance.

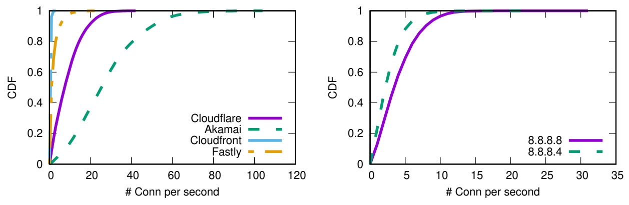

Figure 12: CDF of the number of connections per second for the most-seen IPs in each CDN and DNS service.

Resource (usage unit)	Tofino 1 [34]
Stages (number)	100% I + 8% E
Hash Dist Unit (avg.)	48.6%
Logical Table ID (avg.)	34.9%
SRAM (avg.)	4.3%
TCAM (avg.)	0.0%
Exact Match Input Xbar (avg.)	11.3%
Ternary Match Input Xbar (avg.)	0.0%

Table 6: Data plane resource usage in Tofino 1.

overhead while maximizing privacy. We leave optimizing this migration strategy as future work.

A.4 Notes on the constant-migration result

The reconstruction attacks follow a binomial distribution, and the number of correctly reconstructed sub-connections is expected to be $n' * (1-e)$, when the number of tested sub-connections n' is large enough. Decreasing k makes sub-connections shorter, but also increases n'. Therefore, the lengths of the reconstructed connections could be approximately the same under different k. Quantitatively, the connections reconstructed under different k have similar information leakage (6.64 bits) as analyzed by WeFDE [40, 50].

A.5 RAVEN's data plane resource usage

Table 6 shows the data plane resource usage when running RAVEN's P4$_{16}$ program. Not surprisingly, RAVEN uses a noticeable amount of hash units and logical tables for performing multiple hashing, substitution, and permutation operations. Still, the utilization is less than 50% for each, leaving enough room for other applications or operations if needed. RAVEN uses all stages in the ingress pipeline, but that simply indicates the total pipeline length of the RAVEN program alone; other *independent* applications may be able to share stages with RAVEN. Besides, the egress pipeline is barely utilized by RAVEN.

Encrypting a 64-bit plaintext requires 64-bit keys. We maintain three versions of key sets for each client group, where each key set has three keys. We create eight client groups based on the lowest three bits of destination IP address. The number of resulting 64-bit keys is 72 (3x3x8), which is trivial in terms of memory usage.

A.6 RAVEN extensions

Protecting hierarchical IPv6 addresses. In § 3 we focus on the network that has no subnets. In fact, the same encryption scheme can be applied to protect hierarchical IPv6 addresses from leaking internal routing details. The encryption takes the client ID, port, padding, and the subnet portion of the address as input, and the subnet portion of the address will also be used for storing ciphertext. More specifically, given an address $IP = P \,||\, R \,||\, C$, where $P = P1 \,||\, P2$ ($P1$ is the network prefix, $P2$ is the subnet prefix, R is the reversed field and C is the client bits), RAVEN encrypts the address as $E(P2 \,||\, C \,||\, port \,||\, padding)$, and $|P2|$ bits of the ciphertext are stored in $P2$. The adversary can only learn $P1$ from the encrypted address but nothing about subnets.

Protecting IPv4 networks. We can also use RAVEN to protect IPv4 networks that have IPv6 connectivity. The encryption scheme remains unchanged and clients' IPv4 addresses are used as client IDs during encryption. The switch only needs to maintain additional mapping tables for converting destination IPv4 addresses to the corresponding IPv6 addresses (and backward). We would like to point out due to IPv4 IP-IDs, RAVEN can only guarantee **P1** and **P2**, unless both ends use random IP-IDs.

Connectionless protocols: DNS and NTP. When using random padding, RAVEN is able to achieve per-packet encryption. All outgoing packets will have distinct IP addresses. This is useful for *connectionless* protocols such as DNS and NTP. Conventional DNS (Do53) and NTP reveal the client IP addresses, which are known to cause privacy and security issues. For instance, the adversary may use DNS information to infer users' browsing behaviors, fingerprint and track users, or even deanonymize Tor users [26, 30, 72]; malicious public NTP servers can gather client IPv4 and IPv6 addresses to find targets for unauthorized scans [56]. For Do53 and NTP, RAVEN can provide per-packet IP rotation to hide client IP addresses at the IP layer, requiring no client/server-side cooperation, nor protocol modification. Moreover, RAVEN is compatible with certain DNS protection mechanisms such as DNSCrypt [17] and DNS-over-QUIC (DoQ) [33].

WireGuard. We implemented a simple prototype to validate that RAVEN can work with WireGuard. We rotated the encryption key every 1 second and download 100 randomly selected files with varying sizes (1 KB to 10 GB) from two websites [46, 60]. All files download successfully through WireGuard and RAVEN, and the SHA1 hash of every file matches that of equivalent downloads directly. We repeated the same test under per-packet IP encryption (by using per-packet random bits as the padding), and WireGuard can still work correctly. Throughput tests suggest the client throughput drops by about 20% when doing per-packet encryption compared to no encryption. RAVEN with unmodified WireGuard can easily guarantee **P1** and **P2**. For **P3**, we need to rotate the peer identification in a similar manner as we do in QUIC.