MIMIQ: Masking IPs with Migration in QUIC

Yashodhar Govil, Liang Wang, and Jennifer Rexford
Princeton University

Abstract

The emerging QUIC transport protocol offers new opportunities to protect user privacy. We present MIMIQ, a privacy-enhancing system that leverages QUIC to protect user identity and thwart traffic-analysis attacks. MIMIQ leverages QUIC's connection migration capability to change a client's IP address frequently—even *within* individual connections—without disrupting ongoing transfers or changing the client's physical location. MIMIQ is readily deployable, requiring no cooperation from networks other than the trusted network where it runs. The trusted network facilitates routing of return traffic by running an address allocation server that assigns IP addresses to clients and forwarding rules to switches. By strategically choosing migration times, MIMIQ can defeat certain traffic-analysis attacks while incurring low performance overhead.

1 Introduction

As the general public has grown more and more concerned with maintaining privacy online, there has been an increasing adoption of encryption over the Internet. Nowadays, almost all communication over the Internet is encrypted [19]. While this may obscure much of the information content, users' Internet Protocol (IP) addresses are still visible to eavesdroppers, and may leak sensitive information [8, 9, 11, 13, 23, 34].

Most existing approaches for hiding users' IP addresses, such as Tor [7] and network-layer anonymity systems [3–5, 15, 30], face serious deployment challenges, because they require either extensive modifications to the Internet or support from many stakeholders (e.g., Autonomous Systems along the client-server path). Alternatively, a single ISP could deploy the Address Hiding Protocol (AHP) [26] to encrypt each flow's source IP address to make it look like another random address in the same network (similar to NAT). However, AHP's address mixing is *coarse-grained*, with randomization at the flow level. As such, AHP does not protect against traffic-analysis attacks, which can learn a user's sensitive information based on a single flow [14, 22, 24, 32, 33].

We therefore ask this question: Can we achieve *fine-grained* and *flexible* IP address mixing? Instead of randomizing a user's IP address on a per-flow basis, we want to randomize the IP address in the middle of a flow, or even every few packets. At first glance, this seems to be infeasible,

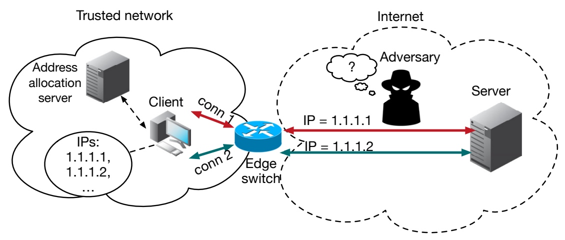

Figure 1: MIMIQ setup: clients in the trusted network communicate with an unmodified QUIC server.

as doing so in TCP, the current cornerstone of the Internet, would break ongoing connections.

The emerging QUIC [18] transport protocol, however, brings a new opportunity—a connection migration feature that can change a client's IP address during an ongoing connection. Originally designed to support mobile clients, connection migration can be used in a novel way to randomize client IP addresses. We propose MIMIQ (**M**asking **IP**s with **M**igration **In** **Q**UIC), a privacy-enhancing system that enables flexible IP address mixing. MIMIQ only relies on a single trusted network for deployment, as shown in Figure 1. Using MIMIQ, a client frequently changes its IP address (i.e., IP hopping) within the trusted network's address space, without changing locations or disrupting ongoing connections. Leveraging IP hopping and QUIC, MIMIQ prevents an adversary from discovering the client originating a flow or associating multiple flows with the same client. By changing the IP address in the middle of a connection, MIMIQ can split a connection into multiple smaller flows to reduce the amount of information the adversary can learn from a single connection, further mitigating traffic-analysis attacks.

One major challenge is that changing the client's IP address makes it difficult to route return traffic back to the client. Another challenge is to avoid different clients using the same IP address at the same time (i.e., address collisions) to prevent a client from receiving traffic intended for another client. To handle these issues, the trusted network runs an address allocation server that assigns IP addresses to its clients and forwarding-table entries to its switches. Before performing connection migration, a client requests new IP addresses from the allocation server, which can carefully select addresses to avoid collisions while preventing the adversary from guessing the IP addresses assigned to each client. The server then updates the forwarding rules in some critical switches to facilitate routing of return traffic. The cooperation between the allocation server and network switches enables MIMIQ to scale to large networks, which provide larger anonymity sets (i.e., more clients and more IP addresses) for better privacy. In a way, MIMIQ achieves connection migration for a stationary client by migrating the client *twice*—once in the transport layer with QUIC and again in the routing layer.

MIMIQ is incrementally deployable: other than the allocation server and several critical switches in the trusted network, MIMIQ does not make modifications to the existing network and server infrastructure. We do not require changes to QUIC-based client applications or the remote server (beyond running QUIC), or require the participation of other networks. The low deployment barriers allow access ISPs to participate for financial incentives, e.g., an ISP can deploy MIMIQ as a privacy-enhancing service to attract more customers.

We implement a proof-of-concept prototype of MIMIQ and test its feasibility using Mininet [21] with an unmodified QUIC server. As expected, performance overhead increases with migration frequency. However, clients who prefer privacy over performance may tolerate higher overhead. In fact, clients can apply flexible IP hopping strategies to strike a balance between performance and privacy. We demonstrate that, under certain scenarios, the client only needs to change its IP address once during a connection to effectively defeat the adversary.

Overall, MIMIQ is a novel privacy application of QUIC. Our work suggests that QUIC has great potential for bootstrapping advanced privacy-enhancing techniques.

2 Background

Threat model. A client communicates through a trusted provider (e.g., enterprise or ISP network) with an unmodified remote server, using standard QUIC [17] (see §3.1 for more details) as the transport protocol. The participating client runs a MIMIQ agent, which can configure host IP addresses, to cooperate with the trusted provider to enable frequent IP address hopping. The trusted provider's network (or *trusted network* for simplicity) can be either a IPv4 or IPv6 network. To have a good anonymity set, we assume a considerable number of clients participate in MIMIQ, and the trusted provider reserves at least the same number of public IP addresses for the participating clients.

We focus on a passive adversary, who sits between the trusted provider's network and the server, and can see and record all traffic exchanged between the trusted network and the server. In addition, the adversary may control a few hosts in the trusted network, send arbitrary traffic to the server, and observe the IP addresses of the traffic sent by these colluding hosts. The adversary's main goal is to discover the client's original network-layer identity. Also, the adversary may perform traffic-analysis attacks to learn sensitive information about the client (e.g., the website the client is visiting). The adversary, however, cannot see traffic within the trusted network (which may reduce the anonymity set) or between the non-colluding clients and the trusted provider. The trusted network should be properly protected with best security practices (e.g., setting up IPsec between geographically distributed sub-networks) and the adversary cannot compromise the trusted network.

The server will not collude with the adversary, but may be curious about the client's identity. We do not try to hide that the changing IP addresses are part of the same connection from the server, but do want to hide the client's identity.

Privacy goals. MIMIQ does not aim to achieve privacy properties as strong as those of Tor and other network-layer anonymity systems (e.g., receiver anonymity and location anonymity). By employing ephemeral IP addresses, MIMIQ aims to make discovering the identity (IP address) of the client more difficult and provide stronger resilience against traffic-analysis attacks. To mitigate traffic-analysis attacks, MIMIQ needs to achieve client unlinkabilty, i.e., the adversary cannot reliably link packets to the same client, which QUIC already provides to some extent. Some side-channels (e.g., packet timing) may be exploited to link traffic to clients, but we do not yet consider them in this work. A key factor in achieving these goals is QUIC's handling of encryption and migration, discussed in §3.1.

Related work. Tor [7] and network-layer anonymity systems, such as LAP [15], Dovetail [30], HORNET [3], and PHI [5], provide stronger privacy guarantees than MIMIQ; however, they typically require multiple ASes along an end-to-end path to cooperate in the protocol, which raises their deployment barriers. SPINE [6] performs per-packet encryption to hide endpoints' IP addresses and to hinder traffic-analysis attacks, but requires the cooperation of endpoint ASes. MASQUE [31, Sec 8.1] suggests mitigating traffic-analysis attacks by transferring QUIC connections among a set of proxy servers. Unfortunately, QUIC has limited support for *server*-address migration [17, Sec 9.6]. Address Hiding Protocol (AHP) [26] can conceal client IP address in a single trusted AS, but it lacks flexibility and fails to protect against traffic-analysis attacks. Virtual Private Networks (VPNs), though lightweight, require careful configurations to avoid unintentional IP leaks [25]. Besides, public VPN services may collect client IP addresses [23] and sell this information to advertising companies to facilitate delivering of IP targeting ads [8].

HyWF aims to defeat traffic-analysis attacks by sending traffic over different routing paths via MPTCP so that adversaries only get partial flow information [12]. However, multipathing itself does not offer unlinkability and anonymity against eavesdroppers or endpoints. A powerful adversary could be listening on multiple paths that the client uses, and may still be able to reconstruct connections based on the global data sequence numbers. In MIMIQ, even an adversary that sees *all* of a client's traffic still cannot reliably reconstruct connections. MIMIQ can be used in combination with multipathing to further improve user privacy.

3 MIMIQ Design

MIMIQ hides client identities by repeatedly changing client IP addresses. MIMIQ has three components: the client agent, the address allocation server, and the edge switch. The allocation server, which has a global view of the IP address assignment in the trusted network, assigns unused IP addresses to a client upon request, and dynamically updates forwarding rules in the switches to route return traffic. Running on participating clients' hosts, the MIMIQ agent communicates with the allocation server to request IP addresses, determines when to hop to a new address and which address to use, and performs IP hopping.

3.1 Seamless connection migration via QUIC

One of the major problems with changing a client's IP address is that all the connections established from the client to the server could be disrupted. If a connection to the server is dropped, the client must attempt to reestablish the connection and incur a significant performance penalty. To avoid this, MIMIQ leverages QUIC's *connection migration* capability.

QUIC is a transport protocol developed by Google [18]. Although still in development, QUIC is seeing rapid adoption as it is is the basis of HTTP/3 [2]: QUIC accounted for about 7% of global Internet traffic in 2016 [18]. Since 2016, the number of QUIC-enabled domains has increased by a factor of 40 to more than 600 K, and the number of QUIC-capable IPs has almost tripled to over 1.6 M [28, 29]. The connection migration feature of QUIC allows a client to change its IP address in the middle of a connection without having to reestablish the connection. QUIC connection migration involves sending a relatively small token for path validation [17, Sec 8.2], which offers substantial performance benefits over a protocol like TCP that does not support migration. These benefits are especially pronounced if the client intends to migrate multiple times.

Unlinkability. Many of QUIC's features facilitate mitigation of client linkability during migration. QUIC packets carry public *connection IDs*, which are used by endpoints to associate flows to the same connection during connection migration. To reduce linkability, the IETF QUIC specification [17, Sec 9] specifies that for each connection endpoints should negotiate a sufficient number of connection IDs during handshakes, and an endpoint must not reuse a connection ID when initiating connection migration. New connection IDs will be negotiated before the available IDs are exhausted. A client might further reduce linkability by changing the source UDP port during migration. QUIC is end-to-end encrypted so that eavesdroppers cannot learn the negotiated connection IDs during handshakes. The token sent to validate connection migration is encrypted, and, with appropriate padding, looks indistinguishable from data sent by the client application. Therefore, the adversary cannot determine with certainty whether and when connection migration events happen. As a result, the adversary also cannot distinguish the case where (1) a client is migrating from one IP address to another, from the case where (2) one client is pausing communications from the first address and a second distinct client is starting communications from the second address.

3.2 Assigning IP addresses to multiple clients

To avoid multiple clients having the same IP address at the same time, MIMIQ uses an address allocation server in the trusted network to assign IP addresses. The client's MIMIQ agent periodically requests a new IP address from the allocation server, and later performs IP hopping. The allocation server returns a randomly-selected, unused address from its address pool. With knowledge of all IP addresses currently in use, the allocation server can ensure that there are no address collisions. To enable flexible IP hopping, the allocation server can assign multiple IP addresses—an *IP address set*—to each client. This also reduces the performance overhead introduced by requesting new IP addresses, as the client does not need to communicate with the allocation server for every connection migration event. Upon receiving an IP address set from the allocation server, the MIMIQ agent can perform flexible IP hopping using different strategies, as we discuss in §5.

Triggering connection migration. The MIMIQ agent triggers connection migration by changing the host IP address (e.g., using `ifconfig`). However, simultaneous connections from the client will share the same source IP address, which may be susceptible to flow correlation attacks. To mitigate this issue, the MIMIQ agent can setup multiple virtual NICs associated with different IP addresses, and instructs the QUIC library (which is modified to be able to interact with the agent) being used by client applications to bind a newly-created connection to a specific IP address. The applications do not need to be aware of these operations. We emphasize the single-connection scenario is actually realistic, e.g., a privacy-savvy client may use QUIC-based VPNs or QuicTor [1] to tunnel all traffic over a single connection.

Managing IP address lifetime. To use IP addresses efficiently, especially in IPv4 networks, MIMIQ recycles assigned IP addresses. Each assigned address (or address set) has a lifetime t_{ip}, which is randomly selected from a given range. The client is expected to stop sending packets using an IP address after t_{ip} seconds. The allocation server tracks the lifetimes of IP addresses, and recycles an address by removing the corresponding forwarding rules (see §3.3) from the switches, so packets associated with that IP address can no longer be sent from/to the client.

3.3 Forwarding return traffic to the clients

In this section we discuss how to handle routing during IP hopping. We only consider scaling MIMIQ to large networks of a particular type in this paper, and leave other types of networks as future work. The target network comprises multiple client-facing switches that connect to the clients, and one edge switch that connects to the Internet. All the client-facing switches and the edge switch are controlled by the address allocation server. We assume each client host and switch in the trusted network has a globally unique static MAC address, each switch has an internal IP address for internal routing, and clients include their MAC addresses in requests to the address allocation server. Each client-facing switch is also assigned a unique identifier *SID* by the address allocation server. Similar to a DHCP relay agent, when a client-facing switch forwards an IP assignment request from a client, it adds its own SID to the request so the allocation server can locate it. The other switches in the network perform normal operations as usual.

Reducing performance overhead via passive ARP learning. In addition to MAC learning, the client-facing switch would normally need to run the Address Resolution Protocol (ARP) or the Neighbor Discovery Protocol (NDP) to determine the MAC address associated with an IP address. However, frequent IP hopping would lead to a high rate of ARP requests, degrading network performance significantly. We take a different approach that we call *passive ARP learning* to solve this issue: The allocation server records the assigned IP address(es) and the client's MAC address, and updates the ARP table in the relevant client-facing switch (Figure 2, ②). One security benefit of using passive ARP learning is to mitigate IP spoofing, as only the allocation server can manage the ARP table.

Efficient routing via address encoding. When receiving a return packet, the edge switch needs to forward the packet to the relevant client-facing switch. A naive approach is to record every assigned IP address and internal address of the relevant client-facing switch. This approach can be sufficient for IPv4 networks: we estimate that the state-of-the-art programmable switch can easily handle the routing entries for /9 IPv4 networks, and can handle /8 networks with certain optimizations [20]. However, it may be hard to scale to IPv6 networks because of large routing tables. Instead, for IPv6 networks, we can encode SID directly into the non-prefix bits of an assigned IP address, and the edge switch only needs to maintain a *static* table that maps SID to client-facing switch internal IP address. A generated IP address is a concatenation of the network prefix, SID, and random bits,

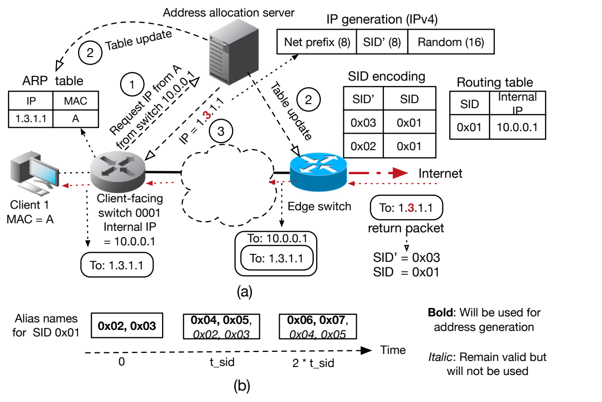

Figure 2: (a) An example of SID encoding. The allocation server encodes SID′ in the assigned IP address. The edge switch extracts SID′ from a return packet, gets the address of the client-facing switch, and sends the encapsulated packet to the switch, who will de-encapsulate the packet and forward it to the client. (b) An example of SID′ updating when k and the size alias name set are 2. The alias names generated at time 0 remain valid for $2 \times t_{\text{sid}}$ seconds.

as shown in Figure 2. One issue is the adversary may be able to link clients to switches based on SID, which reduces the anonymity set. A preliminary solution is the allocation server generates multiple random alias names SID′ for an SID, and randomly encodes one alias name in the address. The alias name sets are updated periodically every t_{sid} seconds.

The edge switch additionally learns (from the allocation server) the mapping between alias names and SID. Given a return packet, the edge switch finds the SID and internal address of the relevant client-facing switch, and encapsulates the packet with an outer IP header in which the destination IP address is set to the address of the switch. The client-facing switch de-encapsulates the packet from the edge switch, and forwards the packet to the client. With the allocation server distributing forwarding rules, MIMIQ can handle asymmetric routing easily in the case that multiple edge switches are deployed.

Assuming l is the number of non-prefix bits in the address, the number of clients that participate in MIMIQ x and the number of client-facing switches y should satisfies $xy \leq 2^l$. This constraint can easily be satisfied for IPv6 networks, considering that IPv6 networks would have large amounts of unused addresses. For instance, MIMIQ can support up to 2^{48} clients and 2^{16} switches for a common /64 IPv6 network. While it is determined by the number of active clients in the naive approach, the forwarding-table size in the encoding approach is determined by the number of client-facing switches, which could be orders of magnitude smaller.

Handling in-flight packets. Return packets in flight might get dropped after migration or SID update. To alleviate this issue, MIMIQ offers mechanisms at the network layer to facilitate the delivery of in-flight packets. The allocation server will keep the forwarding rules for an IP address in the switches for an additional t_{remove} seconds after the IP has expired. Similarly, an alias name set remains valid for at least $k \times t_{\text{sid}}$ seconds, but will not be used for generating new addresses during that period. See Figure 2(b) for an example. The allocation server ensures that during any given $k \times t_{\text{sid}}$ seconds, all valid alias names are unique. If the client interface can have multiple IP addresses at once, packets in flight could still be delivered to the correct host and consumed. As future work, we will develop a method to determine and optimize the timing parameters (e.g., t_{remove} and t_{sid}) in our mechanisms to reduce in-flight packet loss.

3.4 In-network IP generation and hopping

The central allocation server in MIMIQ could become a single point of failure and we therefore look for a way to make MIMIQ more robust. Note that QUIC is also robust to *passive* migration, i.e., NAT rebinding. So alternatively, we can adopt an AHP-like approach to encrypt source IP addresses on-the-fly in the edge switch to trigger connection migration, without using an allocation server. Return traffic handling becomes easier: the edge switch just needs to decrypt the return traffic correctly, and the existing routing infrastructure in the trusted network can remain unmodified. However, with the in-network approach, the edge switch must update QUIC connection IDs during migration; otherwise, the first packet with the new IP address would include an old connection ID that links it directly to existing connections. Since connection IDs are negotiated by the client and the server, the edge switch would need to communicate with the client to get the available connection IDs and also notify the client to request new IDs if the available IDs are exhausted. Not only requiring extensive modifications to the QUIC protocol, this may also affect network performance. Our future work will consider offloading some computations from the allocation server to the edge switch while minimizing the modifications to QUIC.

4 Evaluation

We are still in the process of developing a full prototype. In the evaluation we only demonstrate the feasibility of MIMIQ and estimate the potential performance overhead.

4.1 MIMIQ prototype

To demonstrate feasibility, we implement a proof-of-concept prototype of MIMIQ. [1] We instantiate the address allocation server with an off-the-shelf DHCP server [16]. We set small DHCP lease times, so that old IP address leases expire in time for new requests. Note that one should replace DHCP with customized protocols for better performance. For the

[1]https://github.com/liangw89/p4privacy

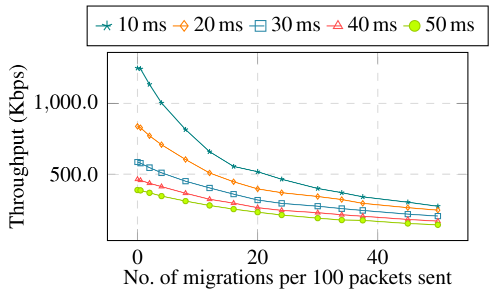

Figure 3: Evaluation of client throughput under different migration frequencies and propagation delays.

core QUIC components, we use the open-source QUIC toy client and server implementations in the Chromium source code [10]. We modify the QUIC client to act as a MIMIQ agent: it sends DHCP requests to the DHCP server, and triggers connection migration by changing its interface IP using ifconfig. The QUIC client can be configured to hop with different frequencies. Ideally, a dedicated MIMIQ agent should run as a daemon and request IP addresses asynchronously for graceful connection migration. Unfortunately, connection migration is not fully supported in the toy client, which causes the client to crash if IP hopping happens before receiving the HTTP response from the server (we can migrate in between successive request-response pairs, though). As a workaround, we let the QUIC client control the IP hopping timing. [2]

Mininet emulation. We use Mininet [21] to emulate two test networks: a *single-client* setting with only one QUIC client running and a *multi-client* setting with four QUIC clients running on different hosts. Each network also has a DHCP server and a QUIC server. In each network, all hosts are connected to a central switch that simply forwards traffic between the hosts. The link capacity is set to 1 Gbps to eliminate any bottleneck in bandwidth.

Feasibility. Each client establishes a QUIC connection with the server and sends multiple `GET` requests. The server responds with the HTML file of `www.example.com`. In between requests, clients contact the DHCP server to acquire new IP addresses. In both settings, the client(s) can migrate at different frequencies without any issues.

4.2 Performance evaluation

We measure the performance overhead of connection migration in a single-client setting. In our experiment, the client sends 1,000 requests over the same connection to the server to download the index.html file of `www.example.com` (totaling about 2.7 MB of data), and hops between two fixed IP addresses after a set number of requests, without contact-

[2]In our tests there are no packets in flight or packet loss. We will investigate the performance impact of packet loss on MIMIQ in the future.

ing the DHCP server. [3] We repeat the experiment with different migration frequencies and round-trip propagation delays (10 ms, 20 ms, 30 ms, 40 ms, and 50 ms), and record the total time taken to complete all requests. To be consistent with previous traffic-analysis studies [24, 32, 33], which usually examine a certain number of packets in a flow, we convert requests to the corresponding number of packets (each client request takes two packets). We define migration frequency as the number of migrations per 100 packets, and consider migration frequencies from 0 (i.e., no migration) to 50 (i.e., every two packets). Note that we perform migration at fixed packet intervals only because it makes it easier to understand the impact of migration on performance; such a predicable migration strategy should not be used in practice.

As expected, more frequent connection migration leads to lower throughput, as shown in Figure 3. The performance penalty becomes less severe as propagation delay increases. For a round-trip time of 50 ms, the throughput drops by only 10% when the migration frequency is four (i.e., migrating per 25 packets). The average latency introduced by each migration also increases as migration frequency increases, ranging from 7 ms to 64 ms, and from 43 ms to 99 ms for round-trip times of 10 ms and 50 ms, respectively. One possible reason for this lies in congestion control. According to the QUIC specification [17], endpoints should reset their congestion control state after a connection migration since the new path may have different congestion properties. Though this affects performance, resetting the congestion state is useful for hiding migration events from the adversary, i.e., the traffic after a migration would look similar to that of a new connection.

A client who prefers privacy over performance may be willing to tolerate higher overhead, but can migrate more strategically to balance the performance/privacy trade-off. See §5 for more discussions.

5 Discussion and Future Work

IP hopping strategy and privacy. The involvement of the client makes MIMIQ more flexible than AHP. Given an IP address set, each client can specify its own policy to freely determine when to hop and which IP address to use. For instance, instead of migration at fixed packet intervals, a client can migrate at random packet intervals or time intervals to make it more difficult to predict migration. One may also migrate more strategically. For instance, if a censor uses the attacks proposed in [33] to detect the application-layer protocols based on encrypted payloads and only looks at the first 30 packets in a flow, the client only needs to migrate once after 30 packets have been exchanged. The censor may block the client's IP if the detected application is blacklisted (e.g., Tor), but by that time the client has already switched to a new IP address and can continue the connection normally. This strategy effectively defeats the censor's attacks with low performance overhead.

Most traffic-analysis attacks (e.g., website fingerprinting attacks and flow correlation attacks) tend to use information extracted from as many packets in a flow as possible to increases attack accuracy [14, 22, 24, 32, 33]. Using MIMIQ to split a connection into smaller flows can reduce the amount information the adversary learns from a single flow, and make traffic-analysis attacks less efficient. We evaluated the efficiency of MIMIQ against state-of-the-art website fingerprinting attacks [27], and found migrating per 25 to 100 packets can reduce attack accuracy to less than 10% in the closed-world setting, which is more efficient than the advanced defenses [12].

Overall, clients should adopt IP hopping strategies based on their own preferences for performance and privacy, and the attacks they want to defeat. Further work is required, however, to inspect the efficiency and performance/privacy trade-off of using MIMIQ to mitigate various traffic-analysis attacks.

Deployment hurdles facing MIMIQ and incentives. To run MIMIQ, no public servers need to be modified, as long as they support QUIC. Since QUIC is rapidly gaining in popularity and will likely be one of the main transport protocols in the future, we believe most servers will soon support QUIC [2,28]. One key element of MIMIQ is that it is aligned with different incentives. The wider Internet has no real incentive to boost the privacy of its users. MIMIQ does not require any changes or effort from these uninterested parties. Instead, the only parties requiring modifications are ASes that have an interest in boosting their privacy by participating in MIMIQ. ISPs may also want to participate because of financial incentives, e.g., an ISP can deploy MIMIQ as a privacy-enhancing service to attract more customers.

6 Conclusion

MIMIQ hides users' identities from downstream ASes and protects against traffic-analysis attacks. Leveraging QUIC's connection migration feature and routing mobility, MIMIQ enables users to change their IP addresses frequently, without changing location or disrupting ongoing transfers. In contrast to prior anonymity solutions, MIMIQ has a low barrier to deployment, as it requires no cooperation from any ASes other than a single trusted network. We demonstrate that MIMIQ is feasible and discuss how MIMIQ can defeat certain traffic-analysis attacks with low performance overhead. As future work, we will implement a prototype that can scale to large networks, conduct a security analysis, and systematically evaluate the effectiveness against various traffic-analysis attacks.

[3] We want to focus on the performance impact of connection migration, but the unmodified DHCP server is a performance bottleneck. The client needs to block ongoing communication and wait for DHCP responses, which adds additional delay that cannot be attributed to migration itself.

References

[1] L. Basyoni, A. Erbad, M. Alsabah, N. Fetais, and M. Guizani. Empirical performance evaluation of QUIC protocol for Tor anonymity network. In *International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 635–642. IEEE, 2019.

[2] M. Bishop. Hypertext transfer protocol version 3 (HTTP/3). https://tools.ietf.org/html/draft-ietf-quic-http-29, 2020.

[3] C. Chen, D. E. Asoni, D. Barrera, G. Danezis, and A. Perrig. Hornet: High-speed onion routing at the network layer. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1441–1454, 2015.

[4] C. Chen, D. E. Asoni, A. Perrig, D. Barrera, G. Danezis, and C. Troncoso. Taranet: Traffic-analysis resistant anonymity at the network layer. In *IEEE European Symposium on Security and Privacy*, pages 137–152. IEEE, 2018.

[5] C. Chen and A. Perrig. PHI: Path-Hidden Lightweight Anonymity Protocol at Network Layer. In *Privacy Enhancing Technologies*, pages 100–117, 2017.

[6] T. Datta, N. Feamster, J. Rexford, and L. Wang. SPINE: Surveillance protection in the network elements. In *USENIX Workshop on Free and Open Communications on the Internet*, 2019.

[7] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security*, 2004.

[8] D. A. Gerken. System and method for selectively acquiring and targeting online advertising based on user IP address, May 20 2008. US Patent 7,376,714.

[9] D. Goodin. Using IPv6 with Linux? You've likely been visited by Shodan and other scanners. https://arstechnica.com/information-technology/2016/02/using-ipv6-with-linux-youve-likely-been-visited-by-shodan-and-other-scanners/, 2016.

[10] Google. Chromium source code. https://chromium.googlesource.com/chromium/src, 2020.

[11] B. Greschbach, T. Pulls, L. M. Roberts, P. Winter, and N. Feamster. The effect of DNS on Tor's anonymity. In *Network and Distributed Systems Security Symposium*, 2017.

[12] S. Henri, G. Garcia-Aviles, P. Serrano, A. Banchs, and P. Thiran. Protecting against website fingerprinting with multihoming. In *Privacy Enhancing Technologies*, volume 2, pages 89–110, 2020.

[13] D. Herrmann, C. Banse, and H. Federrath. Behavior-based tracking: Exploiting characteristic patterns in DNS traffic. *Computers & Security*, 39:17–33, 2013.

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

[15] H.-C. Hsiao, T. H.-J. Kim, A. Perrig, A. Yamada, S. C. Nelson, M. Gruteser, and W. Meng. LAP: Lightweight anonymity and privacy. In *IEEE Symposium on Security and Privacy*, pages 506–520. IEEE, 2012.

[16] Internet Systems Consortium. ISC DHCP. https://www.isc.org/dhcp/, 2020.

[17] J. Iyengar and M. Thomson. QUIC: A UDP-based multiplexed and secure transport. Internet Draft draft-ietf-quic-transport-29, https://www.ietf.org/id/draft-ietf-quic-transport-29.html, 2020.

[18] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, et al. The QUIC transport protocol: Design and internet-scale deployment. In *ACM SIGCOMM*, pages 183–196, 2017.

[19] M. Meeker. Internet trends report, 2019. https://www.bondcap.com/report/itr19/#view/168.

[20] R. Miao, H. Zeng, C. Kim, J. Lee, and M. Yu. Silkroad: Making stateful layer-4 load balancing fast and cheap using switching ASICs. In *ACM SIGCOMM*, pages 15–28, 2017.

[21] Mininet. An instant virtual network on your laptop. http://mininet.org/, 2020.

[22] M. Nasr, A. Bahramali, and A. Houmansadr. Deepcorr. *ACM SIGSAC Conference on Computer and Communications Security*, Jan 2018.

[23] A. O'Driscoll. Does your VPN keep logs? 123 VPN logging policies revealed. https://www.comparitech.com/vpn/vpn-logging-policies/, 2019.

[24] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website fingerprinting in onion routing based anonymization networks. In *ACM Workshop on Privacy in the Electronic Society*, pages 103–114, 2011.

[25] V. C. Perta, M. V. Barbera, G. Tyson, H. Haddadi, and A. Mei. A glance through the VPN looking glass: IPv6 leakage and DNS hijacking in commercial VPN clients. In *Privacy Enhancing Technologies Symposium*, volume 2015, pages 77–91. De Gruyter Open, 2015.

[26] B. Raghavan, T. Kohno, A. C. Snoeren, and D. Wetherall. Enlisting ISPs to improve online privacy: IP address mixing by default. In I. Goldberg and M. J. Atallah, editors, *Privacy Enhancing Technologies Symposium*, pages 143–163, 2009.

[27] M. S. Rahman, P. Sirinam, N. Mathews, K. G. Gangadhara, and M. Wright. Tik-tok: The utility of packet timing in website fingerprinting attacks. In *Privacy Enhancing Technologies Symposium*, 2020.

[28] J. Rüth, I. Poese, C. Dietzel, and O. Hohlfeld. A first look at QUIC in the wild. In R. Beverly, G. Smaragdakis, and A. Feldmann, editors, *Passive and Active Measurement*, pages 255–268. Springer International Publishing, 2018.

[29] J. Rüth, I. Poese, C. Dietzel, and O. Hohlfeld. QUIC research, 2020. https://quic.netray.io/.

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

[31] D. Schinazi. MASQUE obfuscation. https://tools.ietf.org/html/draft-schinazi-masque-obfuscation-02, March 2020.

[32] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal. RAPTOR: Routing attacks on privacy in Tor. In *USENIX Security Symposium*, pages 271–286, 2015.

[33] L. Wang, K. P. Dyer, A. Akella, T. Ristenpart, and T. Shrimpton. Seeing through network-protocol obfuscation. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 57–69, 2015.

[34] F. Zhao, Y. Hori, and K. Sakurai. Analysis of privacy disclosure in DNS query. In *International Conference on Multimedia and Ubiquitous Engineering*, pages 952–957. IEEE, 2007.