IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 5, 2021, accepted February 8, 2021, date of publication February 16, 2021, date of current version February 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3059672

QuicTor: Enhancing Tor for Real-Time Communication Using QUIC Transport Protocol

LAMIAA BASYONI[1], AIMAN ERBAD[2], (Senior Member, IEEE), MASHAEL ALSABAH[3], NOORA FETAIS[4], (Senior Member, IEEE), AMR MOHAMED[4], (Senior Member, IEEE), AND MOHSEN GUIZANI[4], (Fellow, IEEE)

[1]Kindi Research Center, College of Engineering, Qatar University, Doha, Qatar
[2]Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Qatar Foundation, Doha, Qatar
[3]Qatar Computing Research Institute, Hamad Bin Khalifa University, Qatar Foundation, Doha, Qatar
[4]Department of Computer Science and Engineering, College of Engineering, Qatar University, Doha, Qatar

Corresponding author: Aiman Erbad (aerbad@hbku.edu.qa)

This publication was made possible by NPRP grant 7-1469-1-273 from the Qatar National Research Fund (a member of Qatar Foundation). Prof. Mohsen Guizani and Prof. Amr Mohamed's work was supported by Qatar University grant IRCC [2020-003].

ABSTRACT In the past decades, the internet has emerged as the fastest way to access information. However, this revolutionary information age comes with its own set of challenges. The privacy of Internet users is at increasing risk with the advances in surveillance techniques. Users' online behavior, activities, and even personal information are being tracked by ISPs and major tech companies. In response to the increasing need for preserving and protecting the privacy of online users, anonymity networks were developed. Tor anonymity network is a low-latency anonymity network that has gained quite a good reputation over the past years and is being adopted by thousands of users. With the great attention Tor's network is getting, the original design of Tor was proven to have performance limiting issues. With the motivation for addressing the performance limitation in Tor, we present QuicTor, a datagram-based design to solve Tor's transport-layer limiting issue. We evaluated the performance of QuicTor in comparison to vanilla Tor as well as other performance-enhancing proposals. QuicTor achieved significant performance improvements for interactive applications as well as streaming applications. Running Tor over a datagram-based protocol entails a careful security analysis. In this article, we assess the behavior of QuicTor under side-channel attacks aiming to de-anonymize Tor's clients. We show that the performance improvements brought by QuicTor do not reduce the anonymity of clients under the investigated types of attacks.

INDEX TERMS Privacy, anonymity, tor, transport protocols, QUIC.

I. INTRODUCTION

Since its introduction in the 1950s, the internet has revolutionized the landscape of computers and communications on a global scale and is nowadays an integral part of daily lives. The World Wide Web is a client/server application running over the Internet and TCP/IP intranets. Communication over the Internet is highly vulnerable to many attacks threatening the integrity, confidentiality, and authenticity of the traffic. Many of the tasks performed in the digital world require access to the users' private information, which increases the effect of network threats on internet users. Users' credentials and private information are being stolen using Botnet or phishing emails [1], [2]. Advertising agencies acquire users' information from the Internet Service Providers (ISPs), violating users' privacy to develop their marketing strategies. Identity theft is another way of violating users' privacy and causing damage. Some governments monitor internet users to identify and track their opponents, which can threaten their security.

A famous example of the technologies developed to protect the confidentiality and privacy of the users' information, by encrypting the communication between a client and a server, is *Virtual Private Networks (VPNs)*. However,

The associate editor coordinating the review of this manuscript and approving it for publication was Claudio Agostino Ardagna.

in this case, the VPN provider is actually in control of all the traffic and can access it. Moreover, some services are blocking VPNs and cannot be reached through them [3]. Anonymity networks were then introduced to preserve the users' online anonymity. Anonymity networks hide the links between the online user's IP address and his online activities. Many anonymity networks were developed to serve this target, by using multiple proxies between the client and the server. The earliest anonymity networks used based on the technique introduced by Chaum [4] provided anonymity at the expense of high latency to the network [5], [6]. Interactive applications, on the other hand, can not bear such latency, consequently, there was a need for *low-latency* anonymity network.

Tor anonymity network is a low-latency anonymity network that has gained quite a good reputation over the past years and is being adopted by millions of users. Tor was first introduced in 2003, and since then it has been growing in the number of running routers and supported users. As per the statistics from Tor's live network [7], in 2019 the number of directly connected users, not including those connecting through bridges, exceeded 3 million users, the number of operating relays reached 6500 relays and more than 1000 bridge, and more than 60 thousands unique.onion addresses for hidden services.

Tor anonymity network is designed based on the concept of *Onion Routing* [8], [9], to hide the link between the source and destination of TCP traffic. Onion routing provides anonymization of TCP traffic for interactive applications by distributing the traffic over multiple hops as shown in figure 1. The significant increase in the usage of the Tor anonymity network brought to light the fact that, while Tor is powerful in hiding user's identities and protecting their privacy, it suffers from performance issues introducing a delay that could be unacceptable [10], [11].

Achieving an acceptable performance of Tor's anonymity network does not only affect the user's experience, but also the security and anonymity of the network in many ways. Usability is known to be an important factor of security, and the impact of usability on Tor's anonymity was identified in the work of Dingledine and Mathewson [12]. The growth of Tor's network resource, the volunteered relays, is affected by how well it is utilized and how the load-balancing is handled. The growth of Tor's network resources can be directly related to the level of anonymity provided by the network. Several studies addressed the performance problems in Tor [13], [14]. The main goal of these studies was to identify the sources of delay in the network. A clear understanding of the delay causes would lead to a more informed design of Tor's network and help enhance the overall performance. A significant result of these results was that the current design of Tor's transport layer is one of the major sources of delay in the network. Motivated by this knowledge, the Tor community started considering the use of datagram protocols as the base for the transport layer.

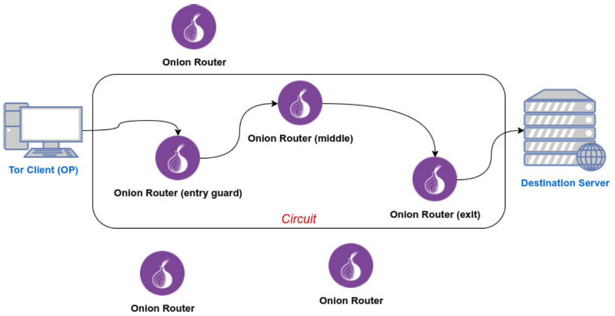


FIGURE 1. Tor network overview.

Since its introduction by Google, the UDP-based protocol QUIC gained increasing attention and is being studied as a possible replacement of TCP for much better performance [15]. One of the motivating goals for designing QUIC was to reduce the delay introduced by TCP's congestion control and flow control mechanisms as well as the delay caused by the handshake process of TLS. The performance gain anticipated by the use of QUIC motivated the idea of running Tor over QUIC. Developers from Tor's community discussed the possibilities of using QUIC in the transport layer of Tor instead of the current design and listed several design decisions to be considered that are specific to the case of Tor's network [16]. In our work, we address the existing problem in Tor's transport layer and expand on the proposed design for running Tor over QUIC in [17]. The work contribution can be summarized as follows.

Our Contribution:

- We discuss in details the features of QUIC protocol that make it a suitable candidate for Tor transport layer.
- We built a realistic test-bed that supports the use of UDP-based protocols (e.g. QUIC) and calibrated our environment using the performance of Tor's live network.
- We evaluated the performance gain of QuicTor over vanilla Tor for different types of applications (web browsing, bulk downloads, and video streaming).
- We present a comprehensive study of the security of QuicTor by analyzing different categories of attacks on the Tor network and highlighting the type of attacks that can be affected by the transport protocol being used. We implemented diverse types of attacks and assessed their impact on QuicTor in comparison to vanilla Tor.

The rest of the article is organized as follows; in Section 2 we present the necessary background of Tor's anonymity network and QUIC protocol. We follow that by reviewing the improvement techniques designed for Tor's network. We discuss QuicTor's design and architecture in section 3. The performance evaluation results are presented and discussed in section 4. In section 5, we present QuicTor's security analysis and evaluation Finally, our conclusion and the plan for future work are in section 6.

II. BACKGROUND

In this section, we start by introducing the basic background of how Tor works and the design of the QUIC protocol, and how it reduces communication delay. Then we review the research work done to enhance the performance and security of the Tor network.

A. TOR

Tor's overlay network consists of several interconnected *Onion Routers (ORs)* over which the traffic is being distributed through *circuits*. On the client-side, the process running is called *Onion Proxy (OP)*. An OP learns the required details to establish a connection to the Tor network by contacting the authority directory routers to obtain the *router descriptors*. A router descriptor is a summary created for each OR, which includes its encryption keys. OPs establish connections to the first *(entry)* OR requesting the build of a circuit to the destination. The entry OR then *extends* the circuit to the next hop, until it reaches the *exit* OR. Figure 2 illustrates how Tor builds its circuits until the client is connected to the destination.

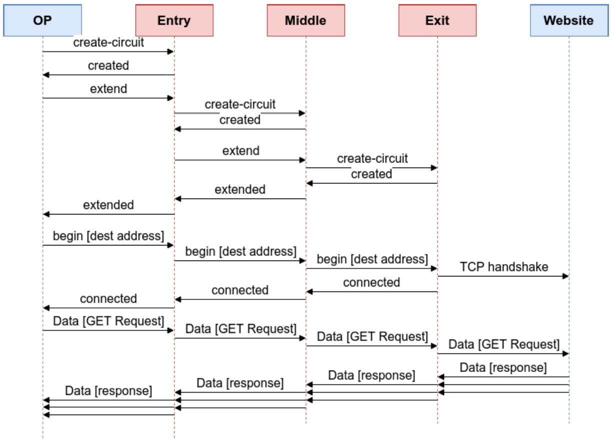


FIGURE 2. Tor's circuit build.

The onion proxy accepts TCP requests (TCP *streams*) and then multiplexes them over the created circuits. The traffic from the client is encrypted with three keys, one for each hop on the circuit. At the exit OR, the destination address is revealed so the exit OR can complete the connection. The original *Onion Routing* design uses one circuit per TCP stream. However, due to the latency cost of this approach, Tor is multiplexing multiple TCP streams over the same circuit. All communication between each two onion routers is done over a TLS connection. The use of TLS adds one more level of encryption and integrity protection to the communication. Tor's communications use a fixed size *cell*, the cell size is 512 byte. The idea of using a fixed cell size is to add some resistance to some types of attacks, such as traffic analysis. However, it was found to be inefficient and results in a distinctive distribution of the packet-size in a specific stream [18]. Hence, control and padding cells are used with variable length to limit the information leak. The typical structure of the cell is shown in figure 3. The circuit ID and command fields are not encrypted, hence it can be processed by all ORs along the circuit to allocate the cell to the corresponding circuit queue. The remaining fields of the cell are encrypted, and can only be processed at the exit OR. The entire cell is then encapsulated in the payload of the transport packet to be sent over the Internet. [18]

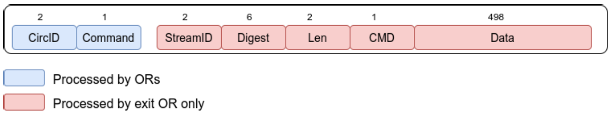


FIGURE 3. Tor's cell.

B. THE CURRENT PROBLEM

Currently, Tor suffers from many performance problems related to network capacity, path selection, queuing, congestion control, and others. While many of them are well-known problems in the networking community, there is currently no perfect solution. In this work, we specifically address issues of Tor's current transport layer design, such as head-of-line blocking which is explained below.

1) HEAD-OF-LINE BLOCKING

The head-of-line blocking problem has been well studied in the area of router design. In essence, this problem stems from the conflicting requirements of multiplexing streams onto a single connection and preserving the ordering of the combined stream in case of failure or packet loss. Head-of-line blocking is an issue related to the use of reliable transport protocols such as TCP. As illustrated in figure 4, this problem happens when a certain TCP flow loses a packet and requires a re-transmission. All subsequent packet of this flows as well as other flows over the same connection are blocked until the lost packet is recovered. In the Tor context, we can map each stream on the connection to a Tor stream that is passing over a circuit, and the TCP connection is the one maintained between two ORs on this circuit. As Tor becomes more popular, we will likely observe similar situations where a file download stream happens to share a TCP connection with a web browsing stream. Moreover, when network links become more congested due to limited capacity, we expect congestion-induced packet losses to become more common, which will lead to more occurrences of the head-of-line blocking problem. Reardon, *et al.* [19] measured the effect of packet dropping on shared TCP Connection. Reardon's experiments concluded that multiplexing circuits over

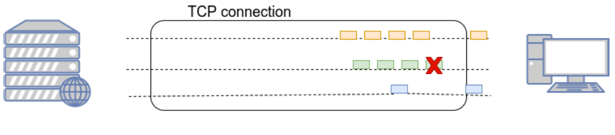


FIGURE 4. Head-of-line-blocking Problem Illustration.

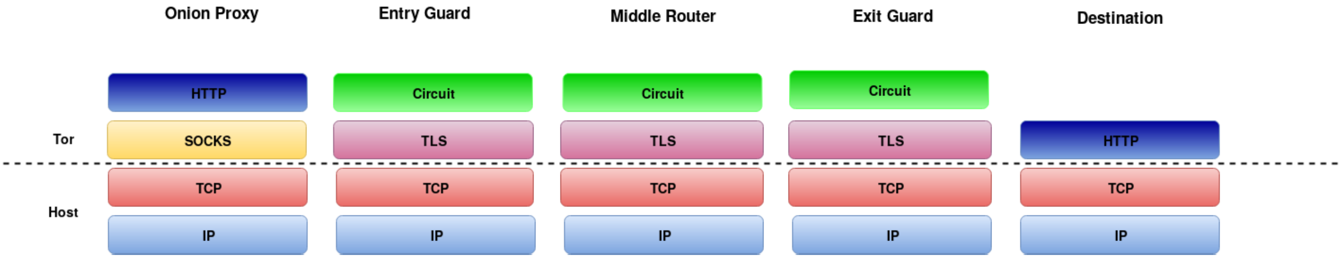


FIGURE 5. **Tor's protocol stack.**

a single TCP connection adds unnecessary latency and degrade the throughput significantly.

C. ENHANCING TOR's PERFORMANCE

There is a considerable number of research proposals made to improve the performance of the Tor network by addressing multiple design weaknesses. The research proposals of interest for the presented work are the ones addressing the transport layer design issues. For a better understanding of the modifications introduced by researchers to achieve better performance, we will adopt the protocol-stack-like description used by Murdoch [20]. As shown in figure 5, all nodes on the path are relying on the host's operating systems implementation of TCP/IP, which ensures reliable packet delivery between the two communicating parties. The TCP stack provides congestion control and in-order delivery, from one hop to the next. On top of TCP comes Tor's built-in TLS stack to provide data integrity, confidentiality, and authenticity. An additional level of confidentiality is performed by the cryptography process in Tor's circuit. The circuit layer, which is implemented at the application level within Tor, provides de-multiplexing between different circuits using the same TLS connection. On the client-side (Onion Proxy), all requests go through SOCKS5 proxy bound to a specific port number, and then passed to the TCP layer.

There is a considerable number of research works attempting to improve the performance of Tor [13]. Improvement efforts concerning traffic management concentrated on removing the source of delays in the network, either by reducing network congestion, traffic overload, or transport overhead. At its application layer, the lack of congestion control in Tor was a major aspect considered by researchers for possible improvements. Proposal for *circuit scheduling* improvement was introduced by Tang and Goldberg [21]. Their method to schedule the packets based on the circuit activity is called *Exponential Weighted Moving Average (EWMA)*. In this method, each circuit keeps a state variable to track the value of the weighted moving average, this value is an indication of how active is this circuit, the less active circuits are then given higher priority in scheduling. Tschorsch and Scheuermann [22] noticed that Tor assigns equal bandwidth for all connections opened between routers, this leads to unfair queuing. They propose to re-allocate any un-utilized bandwidth to a connection that needs more bandwidth. Alsabah, *et al.* [23], introduced a congestion control method for Tor (Tor N23). Tor N23 is based on the algorithm used for Asynchronous Transfer Mode (ATM) Networks.

At the circuit-level, a different approach to enhancing the performance of Tor's network by changing its transport design addresses the multiplexing of several circuits over a single TCP channel. Alsabah and Goldberg [24] proposed the use of a single TCP connection per-circuit. To provide security, they used IPsec and its Encapsulation Security Payload protocol. Although the performance enhancement of PCTCP was significant, the use of IPsec with Tor faces many challenges. Gopal and Heninger [25] in their Torchestra proposed to use two separate TCP connections between each pair of communicating relays. One connection is dedicated to light-weight traffic, the other connection is used for bulk traffic. Torchestra was not tested on a large enough network to get a better understanding of how it improved Tor's performance. It was pointed out by Geddes, *et al.* [26], that PCTCP and Torchestra were subject to socket exhaustion attacks, hence they introduced their IMUX design. IMUX uses a manager and scheduler for the connections.

Aside from that, the process of circuit building and path selection is another rich field for improvements. Barton, *et al.* [27] presented a path-selection algorithm that avoids highly congested relays dynamically while building the circuits. Their algorithm uses a Random Forest classifier to predict the performance of the path and only choose relays with high performance. The authors of [28] propose an extension to Tor's path selection algorithm. They use the latency as a measure of congestion and infer the congestion status of the relays. Based on the congestion information of the network relays, Tor selects the circuit relays that are less congested.

Using datagram protocols for Tor's transport layer was first proposed by Liberatore [29]. Liberatore proposed an extension for the basic specifications of Tor by building the circuits on top of DTLS/UDP. The proposed approach did not offer an alternative to the reliability and in-order delivery functionalities of TCP. The lack of reliability raised two main problems in Liberatore's design. First, the encryption done by Tor at the level of relays was done using the counter mode, in which each block being encrypted depended on the previous and next ones. Hence, in the case of a lost packet, the decryption process would not be successful. Second, the integrity check at both ends of Tor's communication is based on the assumption that no packet will be lost. The extension was meant to be used in parallel to the original design of Tor. The control cells are sent over the TCP connection, only the UDP payload cells are sent over the UDP connection. Eventually, Liberatore's extension did not go any further due to its problems.

Later on, Reardon and Goldberg [19] proposed an improved design using TCP over DTLS. In this design, TCP is moved to the user-level while using Datagram Transport Security Layer (DTLS) to secure the communication between ORs only. Each circuit is assigned a separate user-level TCP connection. The reliability and congestion control are done hop-by-hop. However, the use of user-level TCP suffers from several limitations such as CPU cost. UDP-OR is another approach to improve Tor's performance was proposed by Viecco [30]. Viecco used UDP for the communication between the ORs only, while the end connections at the OP and exit are using TCP. Viecco's design simplifies the processing of packets at intermediate routers, however, it does not provide reliability and in-order delivery functionalities at the routers, which will affect Tor's cryptography and integrity validation. The *head-of-line blocking* problem rises from the fact that if one packet is lost on one TCP streams all other streams are blocked until this lost packet is being resent. To address this problem Nowlan, *et al.* [31], introduced uTor. In uTor, Un-ordered TCP (uTCP) is used for communication between Tor's node and is protected by Un-ordered TLS (uTLS). This allowed TCP to send any available data regardless of the lost packet event. This design adds to the application layer the additional cost of processing the packets, which affects the overall performance of the network. The evaluation of this design showed an insignificant improvement in the performance. Another approach to use datagram protocol as a base for Tor's transport layer was proposed by Loesing, *et al.* [32], using a modified version of libutp library of Bittorrent. However, the implementation was not mature enough to be evaluated against the performance of vanilla Tor.

D. TOR SECURITY

Performance is closely related to the security and anonymity of Tor. Some attacks such as traffic correlation attacks use the network latency and throughput to reduce the anonymity of the network. The possible impacts of datagram proposals for Tor on its security and anonymity were discussed thoroughly in [33]. Therefore, we are presenting here an analysis of different types of de-anonymization attacks that aim to infer the identity of internet users, even if anonymization techniques are applied. In the following, we review Tor's threat model and techniques proposed to exploit Tor's design flows to launch de-anonymization attacks.

1) TOR ADVERSARY MODEL

Most of the de-anonymization attacks assume that the attacker is controlling at least one of the circuit hops, entry or exit guard, or both of them [34], [35]. Furthermore, an attacker can present a compromised client or a malicious destination. An attacker can either passively monitor the traffic, or actively manipulate it. A global attacker can monitor the traffic end-to-end, Tor does not provide security against this type of attackers. A different assumption for an attacker is based on traffic monitoring. The attacker in this model can sniff the network packets and extract their features, train a model, and classify the traffic to identify it. The attacker can also manipulate the packets in a certain way [13], [36], [37]. In Fingerprinting Attacks, the adversary is assumed to be able to monitor the traffic between the client and the entry point to the anonymity network. The adversary then extracts certain features from the traffic, such as packet count, flow direction, the time between consecutive packets. The next step is to match these features to indicative patterns of certain websites, using machine learning techniques. The effectiveness of these attacks depends on the selected features and the machine learning classifier used.

One of the earliest attempts to evaluate the effect of this type of attack on Tor's anonymity network was done by Herrmann, *et al.* [38]. The features they used were the frequency distribution of the size of IP packets, and the classifier used was multinomial Naive Bayes. Herrmann's classifier did not perform well on Tor since it only depended on the packet size, and Tor's cells have fixed size. Later, Panchenko, *et al.* [39], worked on an enhanced version of fingerprinting attack on Tor by choosing different features based on the traffic volume, timing, and direction. Panchenko's classifier reached disturbing results raising red flags for Tor's community. Experimental defenses were recently developed against website fingerprinting attacks on Tor's anonymity network [40]. The AS-level attack is a traffic analysis attack enabled by the presence of the same AS network between the client and the entry guard and between the exit and destination. In their research, Edman and Syverson [41] provide an evaluation of the impact of the AS-level adversary on Tor network security. Their experiment showed that there is a probability of 20% that a single AS appears at the two ends of a circuit. This probability can be reduced by using a different path selection algorithm that is designed to avoid this problem.

2) SIDE CHANNEL ATTACKS

Side-channel attacks are the type of attacks based on some information acquired about the network. In the context of Tor, side-channel attacks can be the first step to launch one of the previously discussed attacks by identifying ORs on the circuit. *Throughput Fingerprinting* is one of the attacks used for this purpose, it depends on the diverse nature of the volunteered routers and their unique behavior while building the circuit to identify the ORs.

Another type of side-channel attack aims to decrease the anonymity of the communication directly, such as *Network Latency*. Two network latency attacks were introduced by Hopper [42], the goal of the first attack was to identify the user initiating the traffic by analyzing the latency distribution of two exit nodes. The second attack aims to locate, approximately, the client by controlling a malicious server that collects any leaked information about the client's network every time the client tries to access the server.

With proposals being made to use datagram-based protocols for Tor's transport layer to improve its performance, an alarming security concern rises on how this type of protocol would affect the security and anonymity of Tor. The study was done by Mathewson and Perry [33] discussed thoroughly the different types of attacks, and specifically the attacks that are more likely to affect Tor over a datagram-based protocol. The described attacks in this study can be viewed as two main types. *First*, attacks exploiting protocol behavioral differences such as re-transmissions, congestion, and flow control. *Second*, attacks exploiting the reduced communication latency, such as timing correlations, and timing watermarking.

Various attacks were developed aiming to reduce the degree of Tor's network anonymity using different network performance metrics such as *latency and throughput* [43].

III. QuicTor

A. QUIC

For decades, TCP has been the key protocol for reliable data transfer over IP networks. However, with the rapid growth of the Internet, many recent applications were designed for interactive use, in which delay is not tolerable. For such applications, TCP was found to be limiting because of its strict in-order-delivery process. TCP is a stream-based protocol, which is suitable for activities carried over a long duration with data that need to be preserved. On the other hand, UDP is more convenient for transactions that need to be executed quickly and independently. For applications that use both short and long transactions, it is difficult to come up with a suitable trade-off that will result in an acceptable performance. In recent years, new transport protocols were designed to provide proper support to different network applications. One possible design approach is to use *unordered* version of TCP (*u*TCP) as a base component and build application-level libraries on top of it [44]. A different design approach is to replace TCP with UDP and implement, at the application level, the required level of reliability. DCCP [45] is a protocol that followed this approach and was designed to provide only the congestion control mechanism to a datagram transport. However, most of these approaches were not widely deployed or used so far.

One recent protocol following the same design approach and is being deployed and used by an increasing number of applications recently is Google's new protocol called QUIC. *Quick UDP Internet Protocol (QUIC)* uses UDP as the transport protocol to avoid the limitations of TCP, and implements at the user-level the congestion and flow control mechanisms.

QUIC was designed with the motivation of reducing communication delay introduced by the handshaking process and by the head-of-line blocking while providing an acceptable level of security and deployability. QUIC is deployed at the user-space to enable its deployment across different platforms. To eliminate the head-of-line blocking issue, QUIC uses an abstracted data structure called *streams* and multiplexes multiple streams within the same connection. QUIC streams represent a reliable bidirectional communication byte-stream. Streams are uniquely identified by stream ID, and the units sent over streams are called *frames*. A QUIC packet, as illustrated in figure 6 is composed of a header and one or more *frames*. After the early handshaking packet exchange, all QUIC packets are fully authenticated and encrypted except for the header parts required for routing and decryption. QUIC implements loss recovery, flow control, and congestion control mechanisms on top of the UDP implementation to ensure reliable transmission.

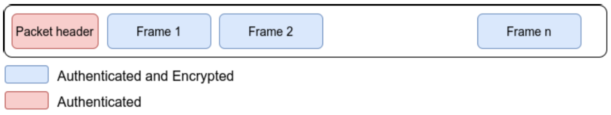


FIGURE 6. QUIC's packet structure.

QUIC avoids the head-of-line blocking problem allowing multiple streams to be transferred over the same connection while ensuring that a lost UDP packet only affects the stream to which it belongs, while other streams can continue to deliver their subsequent packets. Moreover, QUIC limits the buffer space assigned to each specific stream.

1) SECURITY CONCERNS

TCP protocol uses TLS for securing its traffic, which has been proven to provide solid authentication and confidentiality. On the other hand, QUIC protocol is using a different security library, which we discuss in the following. We will explain how QUIC is providing authentication and confidentiality to its traffic. In QUIC all packets are authenticated and encrypted, except for the early negotiation packets and the retry packets. The receiver is always authenticated while the initiator authentication is optional. The authenticating certificate of the receiver party is sent in a *server-config-seg* at the initial handshake phase. The initiator stores the *server-config-seg* received at the connection setup, and use it for later communication. The encryption keys are computed using *Diffie-Hellman(DH)* and are based on the information exchanged during the handshake phase. Tampering with the initial handshaking packets will lead to wrong values of the keys, a reset packet will be sent, unauthenticated and unencrypted, to indicate the failure of the connection [15].

TLS has multiple combination methods of authentication and encryption, the method that was found the most robust is *encrypt-then-authenticate*. In this method, the data is first encrypted then an authentication MAC is computed over the ciphertext. QUIC's data are placed in frames, which are also encrypted and then authenticated within the packets, as shown in figure 6, ensuring a similar level of robustness. TLS is vulnerable to denial-of-service (DoS) attacks, where the attacker imitates a large number of TCP connections and exhausting the server with a large number of handshake requests. By neglecting unauthenticated traffic, QUIC reduces the risks

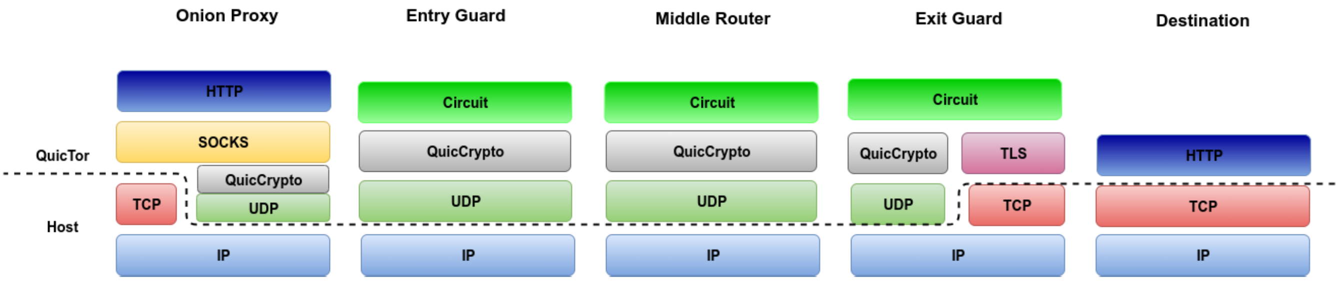


FIGURE 7. QuicTor protocol stack.

for DoS attacks. Both protocols are susceptible to the attacks targeting specific cryptographic standard implemented on the receiver party, such as Bleichenbacher's attack on RSA [46].

B. QuicTor DESIGN

In our work, we built on the direction of using datagram protocol for Tor's transport layer, and we considered the problems faced by previous attempts. QUIC is a UDP-based multiplexed and secure transport protocol designed for bandwidth-hungry and latency-sensitive applications. It was designed by Google and now going through the standardization process in IETF standard track [47]. The main reason QUIC was proposed as a standard way to address head of line blocking at the transport layer in support of HTTP-2. We believe the same issues of the head of line blocking are affecting the performance of TOR. While research proposals such as PCTCP, IMUX, and Torchestra tried to solve the problem at the application level by proposing different methods to de-multiplex Tor's circuits, the use of UDP-based transport protocol, such as QUIC, would provide a solution at the transport-layer level which will also avoid the increasing probability of socket exhausting attacks. Moreover, the use of QUIC protocol for the transport design on Tor's network was considered by Tor's community, as a promising direction to improve the performance of Tor [48]. We believe that QUIC is well-suited to address the two problems mentioned above. First of all, QUIC has native support for multiplexing multiple application-layer streams. This allows QUIC to avoid the head-of-line blocking problem. Besides, QUIC has a pluggable congestion control module whose behavior is specific to each application-layer stream. This means that we can easily change its congestion control behavior for different circuits sharing the same connection in the Tor network. Figure 7 shows QuicTor's protocol stack implementation at each node along the path from the client to the destination. In QuicTor, all communication with onion routers is using QUIC, including the connection from the OP to the entry OR. To explain this design decision, we need to highlight some facts about QUIC and Tor traffic. Since QUIC traffic represents almost 7% of the overall Internet traffic [15], and it is never guaranteed that the destination server will be supporting QUIC protocol, as it is being adopted so far mainly by Google's services. Therefore, we kept the connection between the exit and the destination as it is. It can be seen that the TLS security layer in vanilla tor was replaced by QUIC's security layer *QuicCrypto*. QuicCrypto is part of QUIC that provides transport layer security to a connection. The negotiation of used cryptographic suites is done during the cryptographic handshake which QUIC combines with the transport handshake to reduce initial RTTs. Currently, QUIC is being drafted by IETF and efforts are being made to move the cryptographic handshake implementation to be similar to TLS 1.3 [47]. Two important works have analyzed the security of QUIC [49], [50]. Both confirm it has reasonable security guarantees. QUIC/HTTP-2 was an inspiration work for TLS 1.3. Current versions of the QUIC standard uses TLS 1.3 using creative designs to maintain QUIC performance advantages.

To explain the QUIC communication process, we will refer to the two communicating ORs as *initiator OR* and *receiving OR*. As previously mentioned, QUIC's functionalities are implemented in user-space, including mechanisms to monitor events on UDP sockets, and timeout alerts. This introduced a considerable challenge for our QuicTor API implementation to maintain an accurate timing method that would trigger QUIC's callbacks while dealing with the asynchronous events for the UDP sockets at user-space. QuicTor's API was packaged as a UNIX socket, which means that using a pooling loop to wait for socket events was not possible. To overcome this obstacle without significant re-writing of the code, a dedicated thread was generated for each UDP socket to process its events using libevent, while handling QUIC's alerts using libevent as well. The main thread communicates with each generated thread using a regular UNIX file descriptor (eventfd), which can be treated by the user as an actual socket.

When the initiator OR opens a connection, it starts a blocking operation to create a UDP socket and complete the handshaking with the receiving OR. Once the handshaking is complete the main thread on the initiator OR generates a separate thread for this UDP socket to maintain the QUIC states' updates and the socket events. The generated thread will return eventfd that will be used to trigger the thread in case of pending reads. The generated thread will be responsible for processing received packets without halting the main thread. On the receiving OR's end, the main thread will be listening for an incoming connection, creates a UDP socket, and generates a thread dedicated for this socket to handle its events.

One advantage of this design is that, since all libevent operations and QUIC states are handled in one thread, there will be no need for synchronizing multiple threads, which reduces the complexity of the implementation. For the few shared data structures, fine-grained locking is being used. A second advantage is that we provide a TCP-like usage by moving all asynchronous events to a background thread away from the main thread, in the same way, the kernel is handling them for TCP. Finally, the interface for the API is a standard UNIX socket interface, which reduces the code changes to port existing Tor implementation.

C. QUIC's DEPLOYMENT AND IMPLEMENTATION IN TOR's NETWORK

Tor's original design layers the network communications as follows, connections and channels describing the communication between two nodes only. Circuits and streams, on the other hand, are end-to-end connections. A stream is maintained between the client and the server and is running on top of a circuit. At each hop, the circuit is mapped to a connection. To use the QUIC API within Tor, we decided to limit the modifications to the connection layer while keeping other layers unchanged. Therefore, we added a flag to indicate whether the connection is *using quic* or it is a regular TCP connection. To allow incremental deployment and giving the option of falling back to TCP at any point, we added a new QUIC socket to be used by quic connections along with the TCP socket created by Tor. The architecture of QuicTor is illustrated in figure 8 in which the different layers of QuicTor compared to the existing Vanilla Tor is shown. In QuicTor, the transport protocol used at the kernel layer is UDP. At user-space-level, the QUIC protocol implements its reliability and flow and congestion control functionalities.

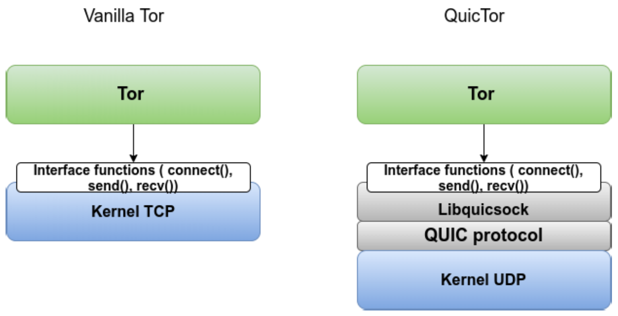

Vanilla Tor
QuicTor
Tor
Interface functions (connect(), send(), recv())
Kernel TCP
Tor
Interface functions (connect(), send(), recv())
Libquicsock
QUIC protocol
Kernel UDP

FIGURE 8. QuicTor Architecture.

On the nodes that support the use of QUIC, all OR connections are being done using the QUIC socket, which includes OR-to-OR connections and OR-to-OP connection. We also simplified the connection layer read and write callbacks by transferring the TLS handshake process to QUIC, which makes it unnecessary to use the handshake code in Tor's callbacks. Other minor modifications, that are not at the connection-level, were required to support the use of QUIC. We needed to add a streamID field for the *packed_cell* structure to be used by QUIC to differentiate user streams. We used the *packed_cell* structure since it's the only one used by ORs for relaying the user streams. The streamID is used whenever Tor calls send to flush some packets to associate these packets with the correct stream. To avoid major modifications in Tor, we designed the QUIC library to provide a similar interface as TCP from Tor's perspective. The API functions connect, send, and recv are following the blocking behavior of TCP while other functions are non-blocking. Moreover, in standard Tor, when a relay is about to send a cell, it will format the cell, copy it to the connection's output buffer, and add a pending write event to the event base. Then in the future when the socket associated with the connection becomes writable, libevent will trigger the write event and run a callback function to send the data out. It is important to note that theoretically, QUIC has no notion of being writable as it uses a non-blocking UDP socket. This means that Tor does not have to wait for buffer space since the buffers are all maintained by QUIC. However, to follow the TCP semantics, we decided to maintain this blocking behavior because we want to make sure that any performance gain comes from QUIC instead of changes in the semantics of TCP.

IV. QuicTor PERFORMANCE EVALUATION

To show the performance gain achieved by the proposed design, we compare the presented work to two other proposed approaches that address the problem of circuit multiplexing over a single TCP connection, namely, we compare our QuicTor to PCTCP [24] and IMUX [26]. However, PCTCP and IMUX address the head-of-line blocking problem at the application layer, by de-multiplexing the circuits and use an appropriate scheduler, while QuicTor addresses the problem at the transport layer. Figure 9 depicts the details of the OR-to-OR connections in QuicTor compared to Vanilla Tor, PCTCP, and IMUX. It can be perceived that QuicTor introduced minimum changes To Tor's architecture by merely adding a different socket identifier to be used for all OR-to-OR connections.

Tor, as a low-latency anonymity network, aims to provide anonymity for the users of interactive web applications such as web browsing. Files downloading, e.g. using *BitTorrent*, is a commonly used application over the web that consumes plenty of its bandwidth [51]. We consider both types of applications in our evaluation of how QuicTor is performing compared to vanilla Tor as well as different enhancement approaches, namely PCTCP and IMUX. In recent years, video streaming has been the top internet application type in terms of traffic percentage. According to the report by Sandvin [51], video streaming reaches 58% of the global downstream traffic. Considering its importance, we evaluate the performance of video streaming over QuicTor compared to vanilla Tor. We implemented our design for QuicTor on Tor's source code version (0.3.3.5-rc).[1] For a fair comparison,

[1] QuicTor's source code is available upon request.

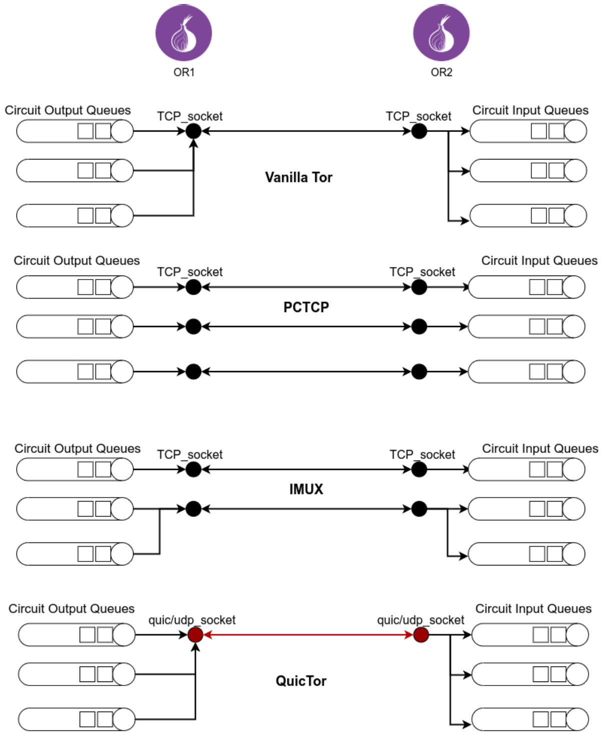


FIGURE 9. OR-to-OR connections in the different approaches.

we ported the implementation of PCTCP and IMUX to the same version. We use a configuration flag to indicate which version of Tor is being used.

We set up our experiments using *NetMirage* [52] network emulator. NetMirage is a platform designed to allow testing IP-based network applications. The feature required in the tested application is the ability to bind to a specific IP address. We had to modify our code to pass the IP provided by NetMirage to the QUIC API for binding instead of using the *localhost* by default.

To ensure a fair comparison, we ported the implementation of both methods to the same version of Tor used in our experiment (0.3.3.5-rc).

A. EXPERIMENT SETUP

NetMirage emulates the network on its *code node* using a GraphML file describing the topology of the network. NetMirage then generates IP addresses for the network nodes on its *edge node(s)* to be assigned to the tested applications. Traffic and communications between applications on the edge node(s) are routed through the core node. The network topology used for NetMirage configuration is in GraphML format, similar to the topologies used by other network simulators such as Shadow [53]. GraphML allows defining network parameters such as *latency, jitters, and drop rate*. In our experiments, we configure NetMirage's core node using the model described by Jansen, *et al.* [54]. NetMirage requires machines to run a Linux-based operating system. We used a machine with Intel Core i7 and 64 GB RAM that runs Ubuntu 16.4 for the NetMirage edge machine. For the core machine, we used an Intel Core i7 powered machine with 8 GB RAM running Ubuntu 16.4 OS. We used a connected graph with each vertex represent a network node to configure the core machine of NetMirage. To simulate real internet behavior, we added latency to the edges that are randomly generated in the range of (50 ms - 100 ms), and drop rate in the range of (1% - 2.5%). The network configuration runs on the edge machine consists of 50 relays and 350 clients. 10% of the clients performed bulk downloads (files of size 5 MB), while the rest of the clients were sending regular HTTP requests representing web browsing activity. Conventionally, the web browsing activity is represented by the download of 320 KB files [24], [26], [55], [56], [25]. However, recently the average size of a web page increased drastically to reach more than 2 MB [57]. Hence, we used files of 2 MB in our experiment to represent web clients. For the video streaming applications performance, we used a 5 minutes video uploaded on a separate server and dedicated one client for video streaming. To validate the realism of our network, we used the performance metrics of Tor's live metrics [7] for 5 MB files, measured over the period starting from 01-11-2019 until 31-01-2020 to calibrate our configuration. The results of Vanilla Tor running on NetMirage's emulated network compared to Tor metrics are shown in figure 10. Tor metrics is an important tool developed by *The Tor Project* to collect data of the live Tor network. The collected data is then aggregated, analyzed, and presented on the Tor metrics website [7]. Tor's relay performance is one of the metrics provided that is used by researchers as a reference for their experiments [56], [58], [59] [24]. The emulated network using NetMirage achieved a performance that is very close to the performance of Tor's live network with an average download time of 10 seconds for 5 MB files, which shows that our emulated network is realistic. We used this network for all of the performance evaluation experiments.

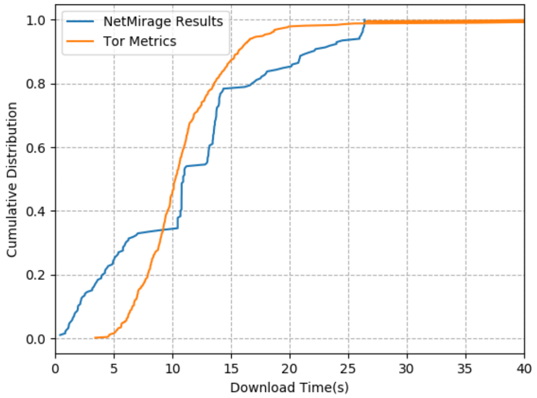
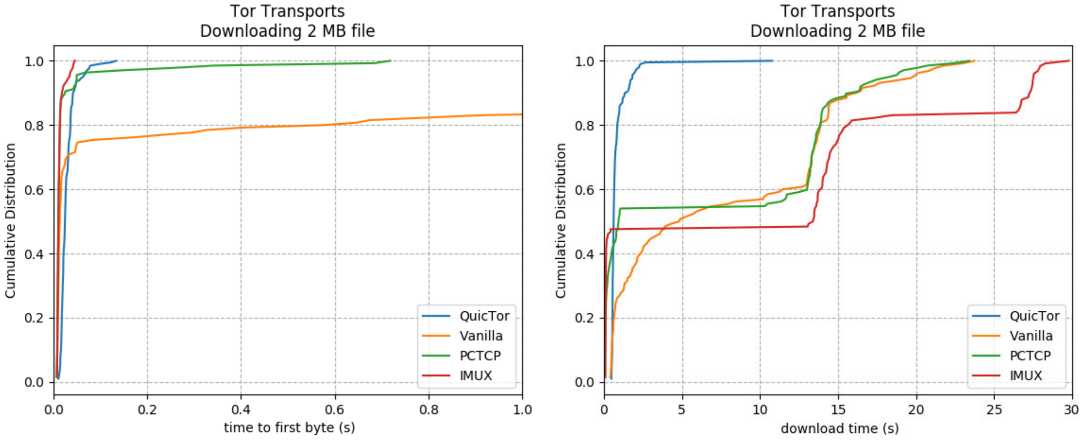


FIGURE 10. Tor network validation.

B. EVALUATION METRICS

For an application like web browsing and file downloading, the time required to complete the action, display the web content, or completely download the file, is the key player in the user's experience, we refer to this metric as



(a) Time to download first byte

(b) Total Time to download file

FIGURE 11. Downloading 2 MB files.

Download Time. In Tor's experiments, the time required to establish the circuit and start receiving the first byte is a considerable factor in its performance evaluation, it will be referred to later as *Time To First Byte*. We use both metrics to evaluate the performance gain of using QuicTor compared to Vanilla Tor.

On the other hand, the user experience of different types of applications such as video streaming is measured by different metrics. In a study of how the quality of experience (QoE) affects the user engagement in video streaming by Dobrian, *et al.* [60], a set of metrics were described to evaluate the QoE for the video streaming applications. Out of the defined metrics, the following metrics are related to network performance.

- **Join Time:** The time required for the player to establish a connection, initialize the playing buffer, and fill the buffer to be able to start playing.
- **Buffering Ratio:** The buffering time as a percent of the total session time. Buffering time is the total time spent filling the playing buffer while the player is frozen.
- **Rate of Buffering Events:** The number of re-buffering events / total session time.

The *session time* is calculated as the total time since the client hits play until the end of the stream. We use these three metrics to evaluate the performance of streaming applications over QuicTor compared to vanilla Tor.

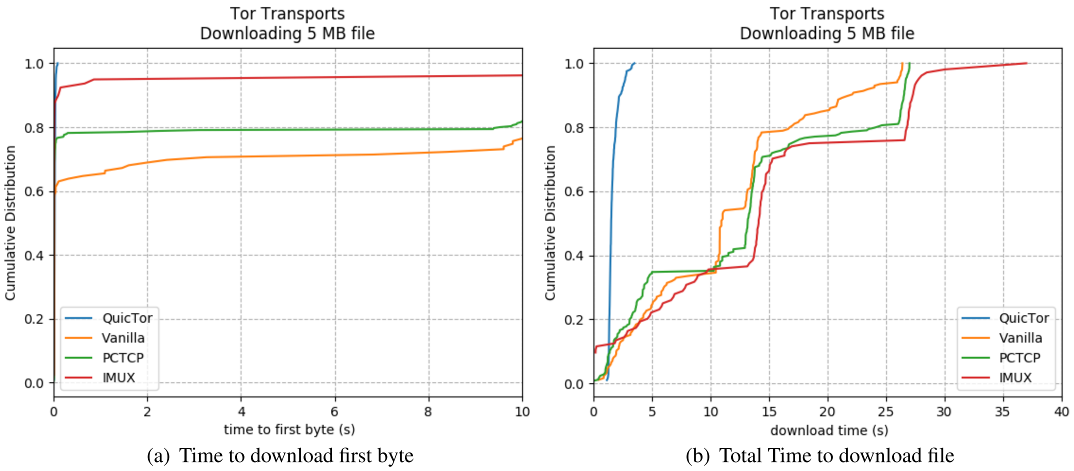
C. RESULTS

The main performance gain from the use of QUIC protocol instead of TLS/TCP lies in *reducing handshaking time* and *overcoming the head-of-line blocking problem*. The number of round trips required for handshaking is the main source of pain for light-weight and short traffic such as web browsing. However, the actual performance gain, in this case, is minimal, equals to two RTTs, and this can be shown by the Time to First Byte results, figure 11-a. In figure 11-b, it can be seen that the average download time of a 2 MB file is reduced by 80% by using QuicTor compared to vanilla Tor. The average download time for PCTCP and IMUX is almost the same as QuicTor, however, The overall performance using QuicTor is improved by 40% compared to PCTCP and IMUX. File sharing applications on the other side last for longer, hence, they can benefit from the improved design of QUIC that eliminated the head-of-line blocking problem. In this case, the actual performance gain of QUIC can be noticed. Figure 12-a shows that 100% of QuicTor requests successfully established the connection in almost 1 second, while only 50% of Tor's connections were established within the same period. For the total time required to complete bulk file download, The average for QuicTor is 3 seconds, and for vanilla, Tor is 15 seconds. QuicTor enhanced the performance for this type of application by almost 80%.

Video streaming applications also benefit from the reduced connection establishment latency of QUIC, which is reflected in the join time (initial buffering duration). It can be seen in figure 13-a that the average initial buffering duration in QuicTor is below 20 seconds, while for vanilla Tor it exceeds 45 seconds. Figure 13-c and 13-b show two ratios that reflect the QoE presented to the user. The rate of buffering events represents how frequent the user will face a frozen player, the less this rate is the better experience the user is getting. The rate of buffering events over QuicTor is 25% less than it is over vanilla Tor. The second ratio is the buffering ratio, which represents the percentage of the session time spent on buffering. QuicTor enhances this metric by 40% compared to vanilla Tor.

V. SECURITY ANALYSIS

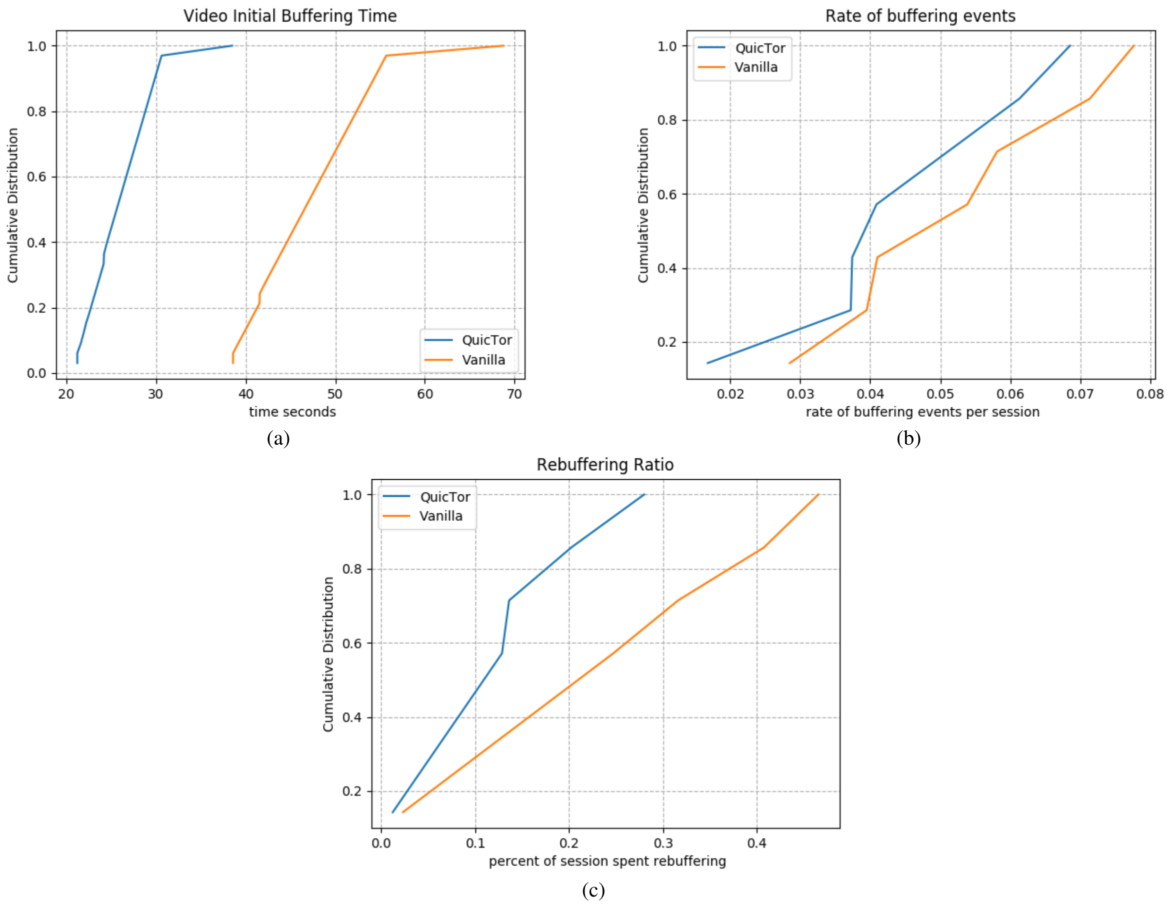
From the aforementioned discussion, we concluded that the category of attacks called *side-channel* attacks use some information gathered from the network traffic, such as delay, and circuit lifetime. The use of different transport protocols could have an impact on the nature of such information, which in turn would either facilitate or impede the launch of



(a) Time to download first byte

(b) Total Time to download file

FIGURE 12. Downloading 5MB files.



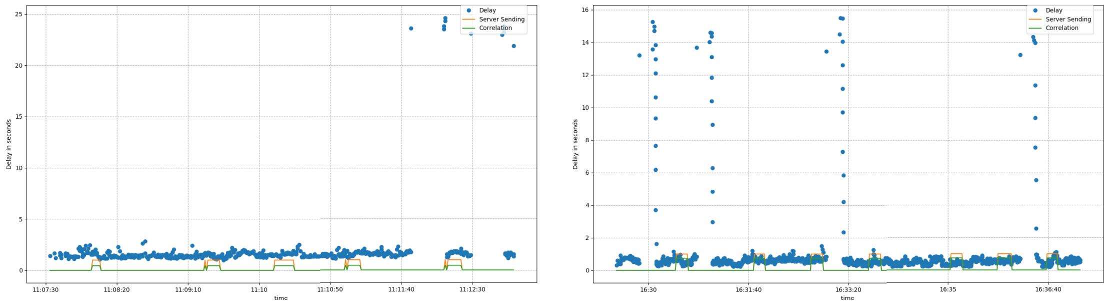
(a)

(b)

(c)

FIGURE 13. Video Streaming Performance Results.

a side-channel attack on Tor's network. The attacks of this category can be *traffic correlation* attacks, or *traffic classification* attacks. In traffic correlation attacks, the adversary monitors the traffic at one end of the connection (entry/exit traffic) as well as at one or more nodes within Tor's network. The target of the adversary is to correlate the entry/exit traffic to the traffic monitored at one or more of Tor's relay to reduce the anonymity of the network. To evaluate the security of QuicTor against side-channel attacks, we implemented two attacks, a timing-based attack described by



(a) True Negative

(b) True Positive

FIGURE 14. Probing Results of Vanilla Tor Relays.

Murdoch and Danezis [61] and a correlation attack described by Mittal, *et al.* [62].

A. LOW-COST TRAFFIC ANALYSIS OF TOR

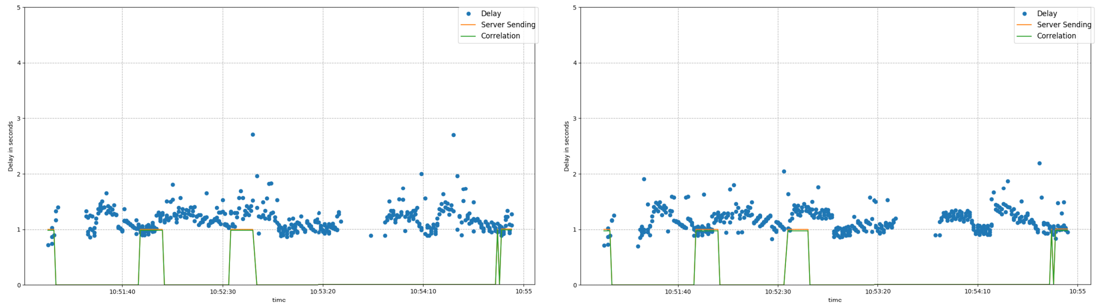
In this attack, [61] Murdoch and Danezis explained how an attacker can launch a traffic correlation attack despite the anonymity property that hides the direct link between communicating parties. Murdoch's attack is based on timing information that the adversary can acquire while staying within the threat model of Tor. The attack depends on the idea that traffic streams over Tor's network have certain characteristics, and that a change in one stream can affect other streams passing through the same node. The adversary assumed in this attack is not global, he cannot observe the timing characteristics of the network. However, the adversary can inject his delay pattern into the network traffic and observe the network streams. The adversary is also assumed to be able to control a corrupted Tor node, which is still within the threat model of Tor. To determine if the injected stream is passing through a specific Tor node, the adversary uses the corrupted node to send a stream to the targeted Tor node and measures the latency of this stream. The adversary then tries to spot the delay pattern injected by the corrupt server in the traffic of the probed relays, and calculate a correlation percentage according to the formula:

$$c = \frac{\sum S(t) * L'(t)}{\sum S(t)} \tag{1}$$

where, S $=$ 1 if the server is sending traffic at time t, and S $=$ 0 otherwise. L'(t) is the normalized latency of the probed Tor relay. In a successful test, the correlation for a true positive (the injected traffic passes through the probed relay) should be higher than the correlation in the case of a true negative (the injected traffic does not pass through the probed relay).

We replicated the experiment described by the authors and tested for vanilla Tor to validate the original implementation. Then, we tried to launch the attack on the QuicTor network to evaluate its behavior against the attack. We created a network topology of 13 relays and 50 clients. One client, the one considered to be the victim, establishes a connection to the malicious server by creating a normal circuit of 3 relays. The malicious server keeps sending for a random period of 15-25 seconds followed by a silent period of 20-40 seconds. For each of the probed relays, a dedicated client is used and configured to allow a single hop. We bind a server to the same IP used for the probed relay and start sending through the client to that server and measure the latency. The rest of the clients are performing regular downloads over the network (web browsing applications). To validate our attack setup, we launched the attack against vanilla Tor and calculated the correlation value for both cases where the probed relay is and is not on the path between the victim client and the malicious server. Figure 14 shows the results from launching the attack on vanilla Tor. In 14-a the probing results of a relay that does not carry the injected traffic by the attacker, while in 14-b the probed relay is on the path from the corrupted server to the victim client. It can be seen from these results that the correlation value is higher in the case of true positive, this indicates a successful test and validates our setup.

The next step was to try launching the attack on QuicTor. Following the same process described for vanilla Tor, we obtained the results shown in figure 15. It was not possible to spot the injected delay pattern in the traffic from all probed relays, whether the relay is on the victim circuit or not. Using logs on QuicTor nodes, we identified the relays on the victim circuit and the relays that are not, the calculated correlation values were almost the same for both cases. The correlation value can be used as an indicator of the impact of the attack on the anonymity of the network. The value of the correlation between the probe data and the victim flow is higher in the cases where the pattern is present in the prob data. Using this information the attacker can significantly reduce the anonymity set by considering the relays with correlation value $\geq$ a certain threshold T. In figure 16, we show the cumulative correlation measured for all probed relays. With correlation threshold $T = 0.4$ [62], it can be seen that the attacker can reduce the anonymity set of the Vanilla Tor network to almost 25% of the total number of relays. On the other hand, the anonymity set of the QuicTor network was not affected. Using the entropy measures defined for measuring anonymity by [63], the attacker can reduce the entropy of the



(a) True Negative

(b) True Positive

FIGURE 15. Probing Results of QuicTor Relays.

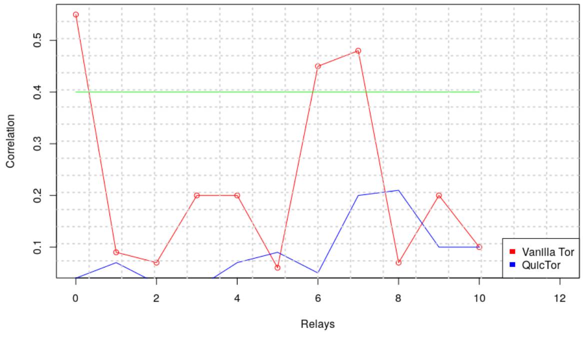


FIGURE 16. Correlation Measured.

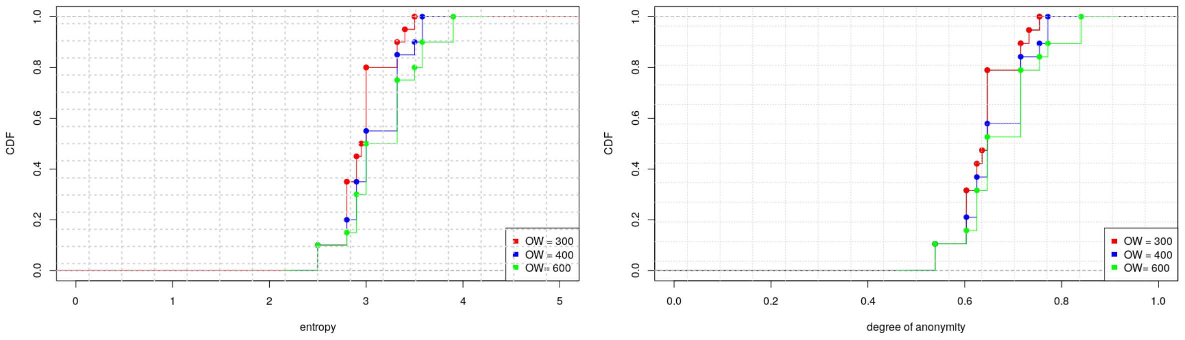
vanilla Tor network by 89%, while for the same correlation threshold the attacker cannot confidently identify any of the relays on the circuits path.

QUIC protocol uses a separate *stream* for every request/response sent to/from the server. Introducing delay in a certain server response will only affect the stream assigned to this response. Other streams for different requests/responses will not be affected by the introduced delay. When the attacker initiates a connection to probe the delay pattern of the relay in question, a new stream is created. The attacker stream in this case does not experience any additional delay. This makes it harder for the attacker to identify whether or not the examined relay is on the circuit path of the victim flow. Based on this, we can claim that timing-based attacks depending on tracking injected delay into the network can not successfully reduce the anonymity of QuicTor. A different attack that also depends on injecting time gapes to be used as a watermark was described by Iacovazzi, *et al.* [64]. Iacovazzi's attack is a flow watermarking attack that aims to de-anonymize Tor's hidden services. Another flow watermarking attack that uses an inter-packet delay pattern as a watermark was introduced by Wang, *et al.* [65].

B. STEALTHY TRAFFIC ANALYSIS USING THROUGHPUT FINGERPRINTING

Mittal's attack [62] is a passive attack that does not require any altering or manipulation of the traffic, instead the attack use the Tor flow's throughput as a fingerprint. The described attacker appears to be like any other Tor user which makes it harder to detect that an attack is being launched. The authors described multiple scenarios to reduce the anonymity of Tor's network by implementing two types of fingerprinting, *stream-based fingerprinting and circuit-based fingerprinting*. *Circuit-based fingerprinting* is used to identify Tor relays, guard relays, and relays offering location hidden services. Mittal's work shows the correlation between the throughput of two circuits in different cases where the circuits share all three relays on the circuit path, two relays shared, and only one relay is common. A conclusion is drawn from these experiments that two circuits with highly correlated throughput have common Tor relay(s). To identify Tor relay(s) along the circuit path of the targeted (*victim*) flow, the attacker is assumed to be able to monitor the victim flow's throughput. Monitoring the flow can be achieved by compromising the exit relay, the destination web server, or the ISP carrying the data. The attacker then probes the throughput of other relays in the network and tries to find a correlation with the throughput measured of the victim flow. To start probing the network relays, the attacker builds a one-hop circuit to these relays. The higher the correlation between the probe flow and the victim the flow, the more probable it is that both flows are traversing through a common relay.

We recreated Mittal's experiments, using 25 relays selected from the network topology we used to configure NetMirage in our previous experiments. We allow our attacker to observe the victim flow for an observing window (OW) of 300 seconds, 400 seconds, and 600 seconds, the observing window represents the lifetime of a client's circuit. In Mittal's experiments, they used a correlation threshold (T) of 0.4 that reflects the moderate confidence of the attacker. However, with QuicTor none of the observed flows correlated higher than 0.3. Using this value as a threshold adds to the uncertainty of the attacker, which further weakens the attack. To quantify the degree of a system's anonymity, entropy is used as a measure [63], [66]. *Entropy* is the level of uncertainty the attacker has about Tor relays in a circuit. After running the attack, the less the entropy is the higher the probability of the attacker being able to identify the circuit relays. In Mittals'



(a) Entropy Set

(b) Degree of Anonymity

FIGURE 17. Entropy Set Reduction Results.

experiment, they were able to reduce the entropy to less than 2.5 bits in 50% of the cases. Given that the maximum entropy for 25 relays is 4.6, the attack was able to reduce the attacker's uncertainty by 40%. Figure 17 depicts the measured entropy after running the experiment for different observation windows. Only 10% of the cases were reduced to 2.5 bits, while 50% of the cases has entropy ≥ 3 bits. In 100% of the cases, the degree of the system's anonymity was ≥ 0.55.

VI. CONCLUSION AND FUTURE WORK

In this work, we presented an assessment of the performance of different applications over QuicTor using a realistic network setup. The results show a significant improvement in the performance of file sharing applications. The performance improvement for video streaming applications would lead to a promising quality of experience for the users. We presented an analysis of the security and anonymity of QuicTor. We found that the basic security guaranteed by TLS/TCP was met by QUIC. The effect of attacks against Tor's anonymity has also been discussed, we reviewed the categories of attacks that are most likely to be affected by the change of the under-laying transport protocol. We implemented two different attacks, evaluated QuicTor's behavior under these attacks, and the results showed that QuicTor's maintained the basic anonymity requirements, and proven better resistance in some cases. The next step in our research is to conduct a study of traffic classification attacks and how they can affect the anonymity of QuicTor. We also plan to address other aspects discussed by Tor's developer Mike Perry in his post [16] regarding the use of Tor over QUIC.

ACKNOWLEDGMENT

Open Access funding provided by the Qatar National Library. The statements made herein are solely the responsibility of the author[s].

REFERENCES

[1] (Nov. 2015). *Pony Botnet Steals Passwords Directly From Your Computer: SSO Alert Priority Moderate*. [Online]. Available: https://www.communications.gov.au/what-we-do/internet/stay-smart-online/alert-service/pony-botnet-steals-passwords-directly-your-computer-sso-alert-priority-moderate

[2] Blueliv. (Oct. 2019). *Credential Theft: The Business Impact of Stolen Credentials*. [Online]. Available: https://www.blueliv.com/cyber-security-and-cyber-threat-intelligence-blog-blueliv/credential-theft/credential-theft-blog-news-and-articles-blueliv/

[3] Admin, FlashRouters, and Joeso. (Dec. 2019). *Why Streaming Services Block VPNS & How to Bypass VPN Blocks*. [Online]. Available: https://blog.flashrouters.com/2018/09/07/why-streaming-services-block-vpns/

[4] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981. [Online]. Available: http://portal.acm.org/citation.cfm?doid=358549.358563

[5] C. Gulcu and G. Tsudik, "Mixing E-mail with babel," in *Proc. Internet Soc. Symp. Netw. Distrib. Syst. Secur.*, Feb. 1996, pp. 2–16.

[6] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *Proc. 19th Int. Conf. Data Eng.*, Jan. 2003, pp. 2–15.

[7] *Welcome to Tor Metrics*. Accessed: Feb. 5, 2020. [Online]. Available: https://metrics.torproject.org/

[8] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous connections and onion routing," in *Proc. IEEE Symp. Secur. Privacy*, May 1997, pp. 44–54.

[9] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding routing information," in *Proc. Int. Workshop Inf. Hiding*. Berlin, Germany: Springer, 1996, pp. 137–150.

[10] K. Loesing, S. J. Murdoch, and R. Dingledine, "A case study on measuring statistical data in the Tor anonymity network," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2010, pp. 203–215.

[11] R. Dingledine and S. J. Murdoch. (2019). *Performance Improvements on Tor or, Why Tor is Slow and What we're Going to do About it*. [Online]. Available: http://www. torproject. org/press/presskit/2009-03-11-performance. pdf

[12] R. Dingledine and N. Mathewson, "Anonymity loves company: Usability and the network effect," in *Proc. WEIS*, 2006, pp. 1–12.

[13] M. Alsabah and I. Goldberg, "Performance and Security Improvements for Tor : A Survey," *ACM Comput. Surv.*, vol. 49, no. 2, pp. 1–41, 2014.

[14] R. Jansen, M. Traudt, J. Geddes, C. Wacek, M. Sherr, and P. Syverson, "KIST: Kernel-informed socket transport for tor," *ACM Trans. Privacy Secur.*, vol. 22, no. 1, pp. 1–37, Jan. 2019.

[15] A. Langley, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, A. Riddoch, W.-T. Chang, Z. Shi, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, and I. Swett, "The QUIC transport protocol: Design and Internet-scale deployment," *Proc. Conf. ACM Special Interest Group Data Commun. SIGCOMM*, 2017, pp. 183–196. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3098822.3098842

[16] M. Perry. (Mar. 2018). *The Case for Tor-Over-Quic*. [Online]. Available: https://lists.torproject.org/pipermail/tor-dev/2018-March/013026.html

[17] L. Basyoni, A. Erbad, M. Alsabah, N. Fetais, and M. Guizani, "Empirical performance evaluation of QUIC protocol for tor anonymity network," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 635–642.

[18] R. Dingledine, N. Mathewson, S. Murdoch, and P. Syverson. (2014). *Tor: The Second-Generation Onion Router (2014 DRAFT v1)*. Cl.Cam.Ac.Uk. [Online]. Available: http://www.cl.cam.ac.uk/~sjm217/papers/tor14design.pdf

[19] J. Reardon and I. Goldberg, "Improving tor using a tcp-over-dtls tunnel," in *Proc. 18th Conf. USENIX Secur. Symp.*, Berkeley, CA, USA: USENIX Association, 2009, pp. 119–134.

[20] S. J. Murdoch, "Comparison of tor datagram designs," Tor Project, Seattle, WA, USA, Tech. Rep. 2011-11-001, 2011.

[21] C. Tang and I. Goldberg, "An improved algorithm for tor circuit scheduling," in *Proc. 17th ACM Conf. Comput. Commun. Secur. CCS*, 2010, pp. 329–339.

[22] F. Tschorsch and B. Scheuermann, "Tor is unfair—And what to do about it," in *Proc. IEEE 36th Conf. Local Comput. Netw.*, Oct. 2011, pp. 432–440.

[23] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. M. Voelker, "Defenestrator: Throwing out windows in tor," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.* Berlin, Germany: Springer, 2011, pp. 134–154.

[24] M. AlSabah and I. Goldberg, "PCTCP: Per-circuit TCP-over-IPsec transport for anonymous communication overlay networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. CCS*, 2013, pp. 349–360. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2508859.2516715

[25] D. Gopal and N. Heninger, "Torchestra: Reducing interactive traffic delays over tor," in *Proc. ACM Workshop Privacy Electron. Soc. WPES*, 2012, pp. 31–42.

[26] J. Geddes, R. Jansen, and N. Hopper, "IMUX: Managing tor connections from two to infinity, and beyond," in *Proc. 13th Workshop Privacy Electron. Soc.*, Nov. 2014, pp. 181–190. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2665943.2665948

[27] A. Barton, M. Imani, J. Ming, and M. Wright, "Towards predicting efficient and anonymous tor circuits," 2018, *arXiv:1805.01977*. [Online]. Available: http://arxiv.org/abs/1805.01977

[28] T. Wang, K. Bauer, C. Forero, and I. Goldberg, "Congestion-aware path selection for tor," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2012, pp. 98–113.

[29] M. Liberatore. (Feb. 2006). *Proposals—Torspec—Tor's Protocol Specifications*. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/proposals/100-tor-spec-udp.txt

[30] C. Viecco, "Udp-or: A fair onion transport design," *Proc. Hot Topics Privacy Enhancing Technol. (HOTPETS'08)*, 2008.

[31] M. F. Nowlan, D. I. Wolinsky, and B. Ford, "Reducing latency in tor circuits with unordered delivery," presented as the 3rd USENIX Workshop Free Open Commun. Internet, 2013.

[32] K. Loesing, S. J. Murdoch, and R. Jansen, "Evaluation of a libutp-based tor datagram implementation," Tor Project, Seattle, WA, USA, Tech. Rep. 2013-10-001, 2013, pp. 1–11.

[33] N. Mathewson and M. Perry, "Towards side channel analysis of datagram tor vs current Tor," Tor Project, Seattle, WA, USA, Tech. Rep. 2018-11-002, 2018, pp. 1–9.

[34] A. Kwon, M. Alsabah, D. Lazar, M. Dacier, and S. Devadas, "Circuit fingerprinting attacks : Passive deanonymization of tor hidden services," in *Proc. USENIX Secur.*, 2015, pp. 287–302.

[35] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against tor," in *Proc. ACM Workshop Privacy Electron. Soc. - WPES*, 2007, pp. 11–20.

[36] P. Mayank and A. K. Singh, "Tor traffic identification," in *Proc. 7th Int. Conf. Commun. Syst. Netw. Technol. (CSNT)*, Nov. 2017, pp. 85–91.

[37] G. He, M. Yang, J. Luo, and X. Gu, "Inferring application type information from tor encrypted traffic," in *Proc. 2nd Int. Conf. Adv. Cloud Big Data*, Nov. 2014, pp. 220–227.

[38] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-Bayes classifier," in *Proc. ACM Workshop Cloud Comput. Secur. - CCSW*, 2009, pp. 31–41.

[39] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proc. 10th Annu. ACM Workshop Privacy Electron. Soc. WPES*, 2011, pp. 103–113.

[40] M. Perry. (2011). *Experimental Defense for Website Traffic Fingerprinting*. [Online]. Available: https://blog.torproject.org/experimental-defense-website-traffic-finger printing

[41] M. Edman and P. Syverson, "As-awareness in tor path selection," in *Proc. 16th ACM Conf. Comput. Commun. Secur. CCS*, 2009, pp. 380–389.

[42] N. Hopper, E. Y. Vasserman, and E. Chan-tin, "How much anonymity does network latency leak," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 2, pp. 1–28, 2010.

[43] J. Geddes, R. Jansen, and N. Hopper, "How low can you go: Balancing performance with anonymity in tor," in *Privacy Enhancing Technologies* (Lecture Notes in Computer Science), vol. 7981. Berlin, Germany: Springer, 2013, pp. 164–184.

[44] M. F. Nowlan, N. Tiwari, J. Iyengar, S. O. Amin, and B. Ford, "Fitting square pegs through round pipes: Unordered delivery wire-compatible with \$*TCP*\$ and \$*TLS*\$," in *Proc. 9th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2012, pp. 383–398.

[45] E. Kohler, M. Handley, and S. Floyd, "Designing DCCP: Congestion control without reliability," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. SIGCOMM*, 2006, pp. 27–38.

[46] T. Jager, "On the security of TLS 1. 3 and QUIC against weaknesses in PKCS # 1 v1. 5 Encryption," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1185–1196.

[47] J. Iyengar and M. Thomson, *Quic: A UDP-Based Multiplexed and Secure Transport; Draft-IETF-QUIC-Transport-24*. Newark, DE, USA: Internet Engineering Task Force, 2019.

[48] M. Perry. (Jul. 2018). *Tor's Open Research Topics: 2018 Edition*. [Online]. Available: https://blog.torproject.org/tors-open-research-topics-2018-edition

[49] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru, "How secure and quick is QUIC? Provable security and performance analyses," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 214–231.

[50] M. Fischlin and F. Günther, "Multi-stage key exchange and the case of Google's QUIC protocol," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 1193–1204.

[51] Sandvine Incorporated ULC. (Oct. 4, 2018). *The Global Internet Phenomena Report: October 2018*. [Online]. Available: https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-repo rt.pdf

[52] N. Unger. (2017). *NetMirage*. [Online]. Available: https://crysp.uwaterloo.ca/software/netmirage/

[53] R. Jansen and N. Hooper, "Shadow: Running tor in a box for accurate and efficient experimentation," Dept. Comput. Sci. Eng., Minnesota Univ Minneapolis, Minneapolis, MI, USA, Tech. Rep. 11-020, 2011.

[54] R. Jansen, K. S. Bauer, N. Hopper, and R. Dingledine, "Methodically modeling the tor network," in *Proc. CSET*, 2012, pp. 1–9.

[55] R. Jansen, P. Syverson, and N. Hopper, "Throttling Tor bandwidth parasites," in *Proc. USENIX Secur. Symp.*, 2012, pp. 349–364.

[56] R. Jansen and M. Traudt, "Tor's been KIST: A case study of transitioning tor research to practice," 2017, *arXiv:1709.01044*. [Online]. Available: http://arxiv.org/abs/1709.01044

[57] Crow, MachMetrics, and StapleCactus. (Jul. 2018). *Website Size: The Average Web Page Size is More Than 2Mb—Twice the Size of the Average Page Just 3 Years Ago*. [Online]. Available: https://www.machmetrics.com/speed-blog/website-size-the-average-web-page-size-is-more-than-2mb-twice-the-size-of-the-average-page-just-3-years-ago/

[58] L. Yang and F. Li, "MTor: A multipath tor routing beyond bandwidth throttling," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Sep. 2015, pp. 479–487.

[59] R. Jansen, M. Juarez, and G. Rafa, "Inside job: Applying traffic analysis to measure tor from within," in *Proc. 25th Symp. Netw. Distrib. Syst. Secur.*, San Diego, CA, USA, Feb. 2018.

[60] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *Proc. ACM SIGCOMM Conf. SIGCOMM - SIGCOMM*, 2011, pp. 362–373.

[61] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of tor," in *Proc. IEEE Symp. Secur. Privacy*, May 2005, pp. 183–195.

[62] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov, "Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting," in *Proc. 18th ACM Conf. Comput. Commun. Secur. CCS*, 2011, pp. 215–226.

[63] D. Claudia, S. Seys, J. Claessens, B. Preneel, and K. U. L. Esat-Cosic, "Towards measuring anonymity," in *Proc. PET 2nd Int. Conf. Privacy Enhancing Technol.*, 2002, pp. 54–68.

[64] A. Iacovazzi, S. Sarda, and Y. Elovici, "Inflow: Inverse network flow watermarking for detecting hidden servers," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 747–755.

[65] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 116–130.

[66] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proc. Int. Workshop Privacy Enhancing Technol.* Berlin, Germany: Springer, 2002, pp. 41–53.

LAMIAA BASYONI received the M.Sc. degree in computer science from the College of Computing and Information Technology, Arab Academy for Science and Technology (AAST), in 2015. In her master's work, she introduced a DCT-based steganography method for 3D models. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Qatar University (QU). She was a Researcher with Dr. Mashael Alsabah and Her research work was addressing performance and security issues in Tor's networks. Her current research interests include privacy and anonymity, networking and protocols, and information security.

AIMAN ERBAD (Senior Member, IEEE) received the B.Sc. degree in computer engineering from the University of Washington, in 2004, the master's degree in computer science in embedded systems and robotics from the University of Essex, U.K., in 2005, and the Ph.D. degree in computer science from The University of British Columbia, Canada, in 2012. He is currently an Associate Professor with the College of Science and Engineering, Hamad Bin Khalifa University (HBKU). His research interests include cloud computing, edge computing, the IoT, private and secure networks, and multimedia systems.

Dr. Erbad received the Platinum Award from H.H. The Emir Sheikh Tamim bin Hamad Al Thani at the Education Excellence Day 2013 (Ph.D. category). He received the 2020 Best Research Paper Award from Computer Communications, IWCMC 2019 Best Paper Award, and IEEE CCWC 2017 Best Paper Award. He serves as an Editor for *KSII Transactions on Internet and Information Systems*, the *International Journal of Sensor Networks* (IJSNet) and served as a Guest Editor for the IEEE Networks.

MASHAEL ALSABAH received the Ph.D. degree in computer science from the University of Waterloo, in 2013. She spent 15 months at MIT as a Visiting Scientist during which she identified a critical security vulnerability that was featured on MIT's front webpage, dozens of news articles, and led to the shutdown of Agora, one of the largest dark web markets. She is currently a Senior Scientist with the Qatar Computing Research Institute. Her research articles are assigned readings in a number of top U.S. universities such as UC Berkeley, and Johns Hopkins University. Her research interest includes the general area of privacy-enhancing technologies with a particular interest in anonymous and secure communication, cryptocurrency, traffic analysis, and side-channel attacks. She received many awards including the Andreas Pfitzmann Best Paper Award at Privacy Enhancing Technologies Symposium and received the Platinum Award for education excellence by Her Highness the Emir.

NOORA FETAIS (Senior Member, IEEE) received the executive master's degree in leadership from Georgetown University, USA, and the Ph.D. degree from the University of Sussex, U.K. She was a member of the first batch of Qatar Leadership Center (Current and Future Leaders Program), from 2011 to 2013. She held various professional and volunteered positions including the Vice Chair of the IEEE-Qatar Section, Qatar Ambassador of Women in Data Science (WiDS), Stanford University, among others. She is currently an Assistant Professor with the Department of Computer Science and Engineering, Qatar University. She is the only woman who chaired the Faculty Senate of Qatar University. She is leading many national and international research teams funded in the field of her specialization. She is also the Founder of the Qatar Women in Cyber Security Middle East (WiCSME) Affiliate Group. Her research interests include visualization for cybersecurity and blockchain. She received a NATO-SPS grant to support CyberWomenWorkshop in Qatar for NATO and partner countries.

AMR MOHAMED (Senior Member, IEEE) received the M.S. and Ph.D. degrees in electrical and computer engineering from The University of British Columbia, Vancouver, Canada, in 2001 and 2006 respectively. He was an Advisory IT Specialist with the IBM Innovation Centre, Vancouver, from 1998 to 2007, taking a leadership role in systems development for vertical industries. He is currently a Professor with the College of Engineering, Qatar University, and also the Director of the Cisco Regional Academy. He has more than 25 years of experience in wireless networking research and industrial systems development. He has authored or coauthored more than 160 refereed journal and conference articles, textbooks, and book chapters in reputable international journals and conferences. His research interests include wireless networking and edge computing for the IoT applications. He has served on the organization committee for many other international conferences as a TPC member, including the IEEE ICC, GLOBECOM, WCNC, LCN, and PIMRC. He holds three awards from IBM Canada for his achievements and leadership and four best paper awards from IEEE conferences. He has served as a Technical Program Committee (TPC) Co-Chair for workshops in IEEEWCNC'16. He has served as a Co-Chair for technical symposia of international conferences, including Globecom'16, Crowncom'15, AICCSA'14, IEEE WLN'11, and IEEE ICT'10. He is serving as a Technical Editor for the *Journal of Internet Technology* and the *International Journal of Sensor Networks*. He has served as a technical reviewer for many international IEEE, ACM, Elsevier, Springer, and Wiley journals.

MOHSEN GUIZANI (Fellow, IEEE) received the B.S. (Hons.) and M.S. degrees in electrical engineering, and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He served as the Associate Vice President for graduate studies with Qatar University, the University of Idaho, Western Michigan University, and the University of West Florida. He also served in academic positions at the University of Missouri-Kansas City, the University of Colorado at Boulder, and Syracuse University. He is currently a Professor with the CSE Department, Qatar University, Qatar. He has authored nine books and more than 500 publications in refereed journals and conferences. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid. He is a Senior Member of ACM. He received the 2017 IEEE Communications Society Recognition Award for his contribution to outstanding research in wireless communications. He received three teaching awards and four research awards throughout his career. He also served as a member, a chair, and a general chair of several international conferences. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He has guest edited a number of special issues in IEEE journals and magazines. He is currently the Editor-in-Chief of the *IEEE Network Magazine*, serves on the editorial boards for several international technical journals and the Founder and the Editor-in-Chief *Wireless Communications and Mobile Computing* journal (Wiley). He served as the IEEE Computer Society Distinguished Speaker, from 2003 to 2005.