Performance and Security Improvements for Tor: A Survey

Mashael AlSabah, Qatar University and Qatar Computing Research Institute
Ian Goldberg, University of Waterloo

Tor [Dingledine et al. 2004] is the most widely used anonymity network today, serving millions of users on a daily basis using a growing number of volunteer-run routers. Since its deployment in 2003, there have been more than three dozen proposals that aim to improve its performance, security, and unobservability. Given the significance of this research area, our goal is to provide the reader with the state of current research directions and challenges in anonymous communication systems, focusing on the Tor network. We shed light on the design weaknesses and challenges facing the network and point out unresolved issues.

1. INTRODUCTION

One of the main challenges facing users on the Internet today is maintaining their privacy, particularly in the face of widespread surveillance [Farrell and Tschofenig 2014]. While the Internet has revolutionized how we communicate and achieve our tasks, its open nature has also attracted many entities that prey on users' personal information for commercial and other uses. Users are increasingly vulnerable to a variety of online threats. Botnets are used to steal login credentials [Kirk 2013], wage Denial of Service (DoS) attacks [Saarinen 2014], and send phishing spam mail [Security 2014]. Internet Service Providers (ISPs) sell their customers' information and clicks to other parties including advertising companies [Blodget 2007; Rushkoff 2012], which collect, aggregate and analyze users' data to strategize their marketing plans at the expense of users' privacy. Worse, users' private data may also be sold to data brokers or identity thieves [Murphy 2012]. Even when entrusted to a government, users' private data are often leaked, stolen [Greenemeier 2006; BBC News 2007; Greenemeier 2008], or exploited by some corrupt government officials to harass activists and others [Christensen 2008].

Anonymity networks have emerged as a solution to allow people to conceal their identities online. This is done by providing *unlinkability* between a user's IP address, his digital fingerprint, and his online activities. Allowing the user to be *anonymous* means that the user is unidentifiable within a set of users [Pfitzmann and Hansen 2008], the *anonymity set*. High-latency anonymity networks are often based on Chaum mixes [Chaum 1981], where multiple proxies (called mixes) are used between the sender and the receiver. When receiving messages, each mix collects a number of them into a batch, intentionally adding delays, before outputting the messages in the batch in a random order to the next mix. This shuffling and delaying is performed in order to defeat *traffic analysis* attacks, where an observer attempts to link the two ends of the communication. Many high-latency anonymity networks have spun off of Chaumian mixes to provide anonymous communication for applications that can tolerate the intentional delays, such as emailing and blogging [Möller et al. 2003; Danezis et al. 2003].

Since much of users' online activities are interactive (e.g., browsing and instant messaging), low-latency anonymity networks have been proposed. Very few such systems have left the realm of academic papers to the Internet, but one key such network is Tor [Dingledine et al. 2004], the most widely used privacy-preserving network, serving millions of people for more than ten years. It grew from a handful of routers—machines that relay users' traffic—to thousands of routers that are operated by volunteers all around the world. Tor is very easy to use; all users have to do is to install the Tor browser bundle, and start browsing the web anonymously. Users of Tor include journalists, activists and whistle-blowers, victims needing online support groups, and other everyday users simply trying to perform private searches and visit websites while trying to protect themselves from being profiled by advertisers. Mixing the diverse traffic of all those users improves the anonymity guarantees provided by the network, as it increases the size of the anonymity set for each user.

Despite its current great potential, Tor has long-term sustainability problems. As more people become more privacy aware [Madden 2014], a future Tor network should have the capacity to serve a significantly larger number of users. Because of several aspects of Tor's design, users currently experience inconvenient performance that manifests itself in the form of large and highly variable delays and download times experienced during web surfing activities. Such delays can be discouraging for users who wish to use it on a daily basis. The challenge to improve the Tor user experience has been taken up by the research community, who have proposed dozens of research proposals and enhancements. Those proposals can be broadly classified to congestion, scalability, routing, and security improvements.

Roadmap. Considering the important role of anonymous communication systems in mitigating the privacy threats we face online, providing an understanding of the research advances in the performance and security of low-latency anonymous communication systems is essential. In this paper, we survey the literature and provide an understanding of the state of current research in this area. We also shed light on the challenges and emerging threats facing those systems. Since Tor is the *de facto* research platform for anonymous communication systems, we focus on Tor for the remainder of this paper.

We start by providing background on low-latency anonymity systems in Section 2, and examine the design of Tor in particular, it being the most widely used and relevant example of such a network. We identify some of the shortcomings of Tor's design in Section 3, and based on the outlined weaknesses, we present our classification of research directions in Section 4. Next, we examine the forefront of this research area by providing a survey on how previous research proposals address Tor's design weaknesses in Sections 5–10. We present ongoing research and unresolved issues in Section 11, and conclude the survey in Section 12.

2. LOW-LATENCY ANONYMITY SYSTEMS

The goal of anonymous communication is to solve the traffic analysis problem, which David Chaum defines as follows: "The problem of keeping confidential who converses with whom, and when they converse." [1981] To that end, anonymous communication systems are designed so that users in the system communicate with their destinations through a single or a number of intermediate hops, where every hop only knows the next and previous hops. Therefore, no hop alone can link the sender with the receiver (unless only a single intermediary hop is used). Messages relayed in the system are generally fixed in size, and they are cryptographically altered at every hop.

Since the introduction of mix-nets in 1981, the area of anonymous communications has evolved into two streams, based on their design and the scope of applications they support: the low-latency and the high-latency anonymity systems.

High-latency anonymity systems, like Mixminion [Danezis et al. 2003], and mix-nets [Chaum 1981] assume a powerful *active global* adversary—one which is able to monitor the input and output links of every node, usually called a mix, in the network. To hide the correspondences between the incoming and the outgoing traffic of a mix, equal-length messages are shuffled, cryptographically altered, and stored for some intentionally added delay before they are sent to the next mix or destination. Because of the additional delay, which can be up to several hours, high-latency anonymity systems can only support delay-tolerant applications, such as e-voting and email.

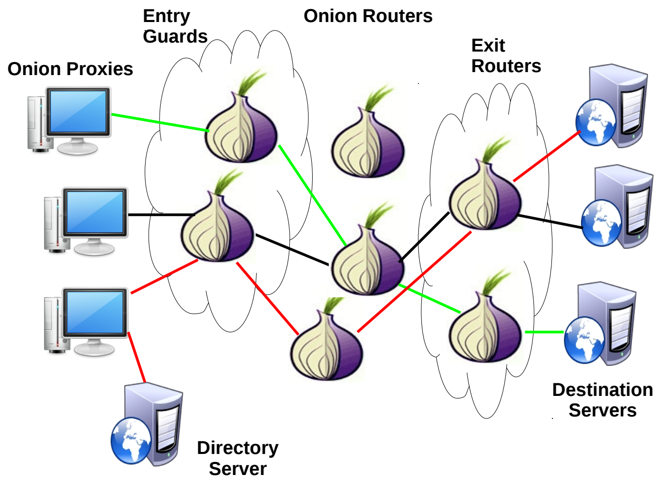

Fig. 1. The Tor network: Clients (running OPs) contact the directory servers periodically to get a list of ORs and their descriptors. Clients then use Tor's router selection algorithm to build circuits. A circuit consists of three hops: entry guard, middle and exit. The exit performs the TCP connection on behalf of the user to the user's destination.

Although high-latency anonymity systems are considered more secure against global eavesdroppers, a family of traffic analysis attacks, known as *statistical disclosure attacks* [Danezis et al. 2007; Mallesh and Wright 2010] has shown its effectiveness in compromising the anonymity of users. In those attacks, the anonymity network is viewed as a black box, and the attacker correlates the traffic that enters and exits the box to determine the communication patterns. Low-latency anonymity systems assume a more relaxed threat model: a *local* adversary who can monitor only part of the network (no more than 20% of the nodes, for example). This threat model restriction exists because a global adversary for a low-latency network can observe (or even induce) traffic timing patterns entering the network, and watch for traffic of a corresponding pattern leaving the network, thus matching sender and recipient. These powerful *traffic correlation attacks* [Murdoch and Danezis 2005; Raymond 2000] are typically easy for a global adversary [Danezis et al. 2007; Mallesh and Wright 2010] and so the literature restricts its attention to local adversaries for the low-latency case. Low-latency anonymity networks are designed to support common interactive applications such as instant messaging, web browsing, and SSH connections, and are far more widely used than their high-latency counterparts. We next present a detailed background on Tor, the most widely used anonymity network, and survey previous proposals that aim to improve it.

2.1. Tor

Tor is a low-latency anonymity network based on the concept of onion routing [Reed et al. 1998]. The network today consists of approximately 6000 volunteer-operated routers [Tor Project 2015a], known as *Onion Routers* (ORs). Each OR creates a *router descriptor* that contains its contact information, such as its IP address, ports, public keys, and its bandwidth capabilities, and sends the descriptor to *directory authorities*. These authorities construct a network *consensus* document, which they send, along with the descriptors, to *directory servers*. Tor clients, nicknamed *Onion Proxies* (OPs),

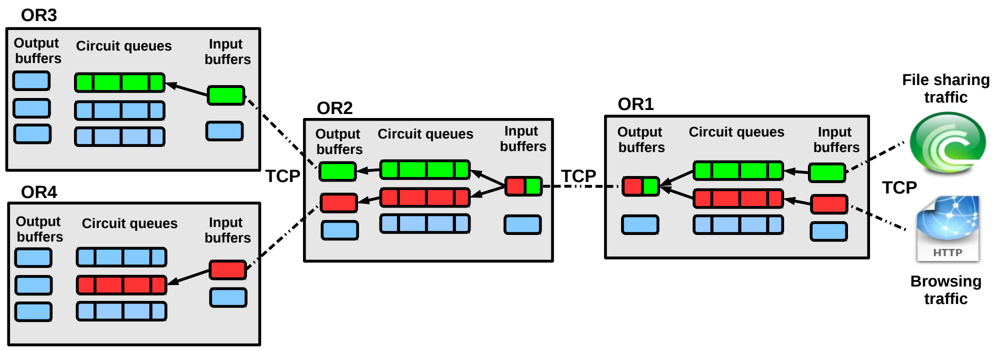

Fig. 2. This figure (adapted from AlSabah and Goldberg [2013]) shows Tor's application-level queueing architecture. Tor maintains input buffers, circuit queues, and output buffers, in addition to the kernel's TCP buffers. This figure also demonstrates the cross-circuit interference problem (see Section 3.5).

download the descriptors and the consensus from the directory servers in order to build paths, referred to as *circuits*, through the network before they can communicate with their Internet destinations. Each circuit usually consists of three ORs, or *hops*, which are referred to as the *entry guard*, *middle*, and *exit* OR, according to their position in the circuit. ORs in a circuit are connected by TCP connections and TLS [Dierks and Rescorla 2008] is used to provide hop-by-hop authenticity, data integrity and confidentiality. Traffic in Tor travels in fixed-sized units (512 bytes) called *cells*. Figure 1 visualizes the Tor network.

2.2. Circuits and Streams

The OP builds circuits by first selecting three routers, X_i, according to Tor's bandwidth-weighted router selection algorithm (described in Section 2.6). Next, to start establishing the circuit, the OP sends a *create_fast* command to X_1, which responds with a *created_fast* reply. To extend the encrypted channel, the OP sends an *extend* command to X_1, containing in its payload a *create* command and the first half of a Diffie-Hellman (DH) handshake for router X_2, encrypted to X_2's public key. Router X_1 forwards this *create* command to router X_2, and when it receives a *created* cell back from router X_2, it forwards its payload in an *extended* cell to the OP to finish the client's DH handshake with router X_2. The same procedure is carried out for each subsequent OR added to the circuit, establishing a shared session key between the OP and each of the routers in the circuit. Cells sent along a circuit by an OP are multiply encrypted, with one layer of encryption (using the above session key) for each hop in the circuit. Each hop decrypts one layer before passing the cell to the next hop.

For performance reasons, an OP preemptively creates a number of spare circuits for its user applications. In order to communicate with the client application (for example, a web browser), the OP exposes a SOCKS proxy, which the browser is configured to use. When the browser connects a new TCP stream to the OP's SOCKS proxy, the OP attaches the new stream to an appropriate pre-established circuit. (OPs can multiplex several TCP *streams* over one circuit.) Once a circuit begins being used, it generally has a lifetime of ten minutes, after which no new streams will be attached to it, though the circuit will continue to exist until all streams it contains are closed.

Note that only the exit node can observe the user's traffic, and only the entry guard knows the identity of the user. If both the entry guard and exit node cooperate, however, they can use traffic analysis to link the initiator to her destination.

2.3. Tor's Queuing Architecture

Tor uses a tiered buffer architecture to manage cells traveling through circuits, as shown in Figure 2. When an OR receives a cell from an external server or from another OR or OP, the cell is passed from the kernel TCP receive buffer to a corresponding 32 KiB connection-level input buffer in Tor. After the cell is encrypted or decrypted, it is placed on the appropriate FIFO circuit queue. Since several circuits share the same connection output buffer, a scheduler is used to retrieve cells from the circuit queues to be placed on a 32 KiB output buffer. Finally, the cells are sent to the kernel TCP send buffer which flushes them to the next OR or OP.

2.4. Circuit Scheduling

Tor uses a label-switching design that multiplexes several circuits across the same Tor routers. In order to ensure that each circuit is given a fair share of the routers' bandwidth, Tor's original design employed a round-robin queuing mechanism. Each circuit is serviced in a first-come, first-served manner, which ensures that each circuit is given a fair share of the available bandwidth. However, McCoy *et al.*[2008] have revealed that the distribution of application traffic on Tor is not uniform across all circuits: a relatively small number of circuits (e.g., bulk file downloaders) consume a disproportional amount of the network's bandwidth. To mitigate the unfairness, Tang and Goldberg [2010] proposed a circuit scheduling prioritization scheme (described in Section 5.2.1) so that interactive circuits are serviced before bulk-downloader circuits. This prioritized circuit scheduling is currently deployed on the live Tor network.

2.5. Traffic Throttling

In order to provide routers with the ability to control and manage congestion and network overload, the Tor network employs several levels of throttling to limit the amount of data that enters the network, as follows:

— Rate limiting: each Tor OR implements a token bucket rate limiting algorithm that limits the amount of bandwidth an OR can spend on the network. The rate limit can be configured by the OR operator.

— Circuit windows: each circuit uses an end-to-end window-based flow control algorithm that limits the amount of data traveling down a circuit at any time. The OP and the exit maintain a window that is initialized to 1000 cells (500 KiB). This means that the end OR (exit or OP) can send 1000 data cells down the circuit before an acknowledgment is received. Every time a data cell is sent, the window size is decremented by 1, and transmission stops if the window size reaches 0. When the receiving end (OP or exit) receives 100 cells, it sends an acknowledgment cell, known as the circuit_sendme cell, to the sending end (exit or OP). When a circuit_sendme cell is received at any end, the OR increments its circuit windows by 100.

— Stream windows: recall that every Tor circuit multiplexes several TCP streams from a user's applications. Each stream is flow controlled using a stream-level window, which operates in a very similar manner to circuit windows, except that the window size is 500 cells (250 KiB). Every time one end OR receives 50 cells, it sends a stream_sendme acknowledgment cell, which has the effect of incrementing the window at the other end by 50.

2.6. Router Selection Algorithm

In the original Tor proposal, ORs are selected uniformly at random for circuit construction. This is based on the idea that random selection allows all routers to be selected with equal probability, which increases the uncertainty of an observer trying to deanonymize the routers used in a particular circuit. However, due to heterogeneity in resources and to protect users from other classes of attacks, the algorithm was later changed to fulfill the following constraints:

(1) No router appears more than once on a path, and no two routers in the same circuit belong to the same class B network (/16 subnet) or the same *family*. Co-administered routers can be marked as belonging to the same family by operators to avoid hindering users' privacy.
(2) Directory authorities assign flags to routers based on their performance, stability and roles in the network. For example, a "fast" flag is given to an OR that is active, and its bandwidth is either in the top $7/8$ of known active routers or at least a minimum defined amount. An OR with a "stable" flag is defined as one whose weighted mean time between failure is at least the median for all known routers.
(3) As of 2006, to defend against some attacks, such as the predecessor [Wright et al. 2004] and locating hidden services [Øverlier and Syverson 2006] attacks, Tor's router selection algorithm was changed so that the entry node is chosen from a subset of nodes known as the entry guards. An entry guard is a node whose weighted fractional uptime is at least the median for active routers, and its bandwidth is at least the median or at least 250 KB/s [Dingledine and Mathewson 2015b]. At time of writing, a set of 3 guards is selected, assigned a validity period between 30–60 days, and used throughout that time for all circuits. However, to limit opportunities for attacks by malicious guards, this design is currently being changed, so that every client uses one entry guard for a longer period of time [Dingledine et al. 2014].
(4) Selecting a subsequent OR on the path is proportional to its offered bandwidth, in order to ensure that more capable routers are chosen more often. If b_i is the bandwidth[1] offered by router i, then router i is chosen with probability $b_i / \sum_{k=1}^{N} b_k$, where N is the total number of routers in the network.

2.7. Tor's Threat Model

The threat model in Tor assumes a local active adversary that can watch or control part of the network (no more than 20%, for example) and can add, delete or modify traffic in the network. The anonymity of a Tor circuit is compromised if the adversary can watch the two ends, the entry and exit, of the circuit. That is, if an adversary controls a fraction f of the network, then the probability of circuit compromise is f^2, which is the probability of controlling the two ends of a circuit. For example, if we assume a powerful adversary controlling 20% of the network resources, then the adversary will be able to deanonymize only 4% of the user circuits. Other than the end-to-end compromise rate, Díaz *et al.*[2002] proposed the *degree of anonymity* as a metric to quantify anonymity for anonymous communication networks in general. Formerly, the size of the anonymity set had been used as an anonymity metric. Without additional information, all users in the anonymity set have an equal probability of being originators of a message; therefore, the larger the anonymity set, the more anonymity users are provided. However, this metric does not take into account that an attacker can assign different probabilities to users as being originators of a message after observing the system for some time. Díaz *et al.* proposed using entropy $H(X)$ as a tool to calculate the degree of anonymity achieved by users of a system.[2] The

[1]The bandwidth value of a router published in the network consensus is calculated by the directory servers based on a router's advertised bandwidth, adjusted by live bandwidth measurements performed by special servers, known as the bandwidth authorities, whose duty is to periodically probe routers and report their measured capacity to the directory servers. These values are then weighted depending on the router's capabilities and position in the circuit.
[2]A similar entropy-based metric was independently proposed by Serjantov and Danezis [2002].

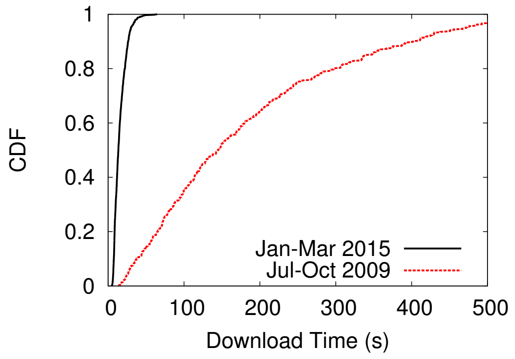

Fig. 3. Download times. The figure compares the download time distributions of 5 MiB files over the live Tor network in 2009 and 2015. At the median, downloading the file completed in around 144 seconds in 2009, and 14 seconds in 2015. Even though the performance has improved since 2009, it is still poor at the fourth quartile where download times ranged from 17 to more than 25 seconds.

entropy is given by $H(X) = -\sum_{i=1}^{N} p_i \log_2(p_i)$, where p_i is a probability mass function corresponding to a sender. If $H_M = log_2(N)$ is the maximum entropy of the system, where N is the size of the anonymity set, then the degree of anonymity d is defined as $d = 1 - \frac{H_M - H(X)}{H_M} = \frac{H(X)}{H_M}$. For the special case of only one user, $d = 0$.

2.8. Anonymity Loves Company

Tor offers anonymity at the expense of intolerable performance costs. Figure 3 shows the download time distribution of 5 MiB files over the Tor network from July to October 2009 [Tor Project 2015b]. During that time period, the median download time was 144 seconds. Over the years, the performance of Tor has improved tremendously, as also shown in Figure 3. The median download time in 2015 is 14 seconds. This improvement could be attributed to many factors such as (1) the increasing capacity and number of routers serving the network, (2) performance improvements that have been adopted over the years, and (3) the fact that Tor was being blocked by different countries, reducing its user base. For example, Tor was blocked in China around the end of September 2009.

Not only do performance problems hinder Tor's wider adoption, but they can have an immense impact on its anonymity [Dingledine and Mathewson 2006]. If users are discouraged from Tor's below-mediocre service, the anonymity set of all users would eventually shrink, which in turn reduces the anonymity guarantees obtained from the network today. Therefore, it is crucial to improve the performance and usability of Tor in order to enhance the anonymity it provides.

2.9. Hidden Services

Tor allows servers to provide content and services over the network while maintaining their anonymity. First, a *hidden service* generates a public/private key pair and chooses some routers randomly as *introduction points*. Then, to advertise its service, the hidden service creates a signed descriptor containing the introduction point information and its own public key. Based on the contents of the descriptor and a validity time period t, a descriptor ID will be generated, and then the descriptor will be published in a Distributed Hash Table (DHT) hash ring that is formed by hidden service directories.

The hidden service computes the directory responsible for holding its descriptor based on a closeness metric between the descriptor ID and the directory's fingerprint, which is the SHA-1 hash of the directory's public key. (At the time of writing, descriptors are uploaded to 6 hidden service directories.) Note that the responsible directories for a specific descriptor change after time t. The hidden service publishes its *onion address*, which is an address of the form abc.onion where abc is a truncated hash of the hidden service public key. The hidden service also maintains circuits to its introduction points and informs them of its public key.

When a client is interested in connecting to hidden service X, it first searches for the latter's onion address `abc.onion`, which can be queried through one of the public routers that is part of the DHT. Next, the client starts preparing for the connection to X by constructing a circuit to a randomly chosen *Rendezvous Point* (RP). The client sends a command cell to the RP containing a 20-byte arbitrary ID to serve as a rendezvous cookie. Then, the client builds a circuit to one of X's introduction points and sends a message (encrypted to X's public key) containing the first half of a DH handshake, along with its RP information, and the rendezvous cookie that was sent to the RP in the previous step. Finally, X builds a circuit to the RP, and the two parties can communicate.

2.10. Blocking Resistance

Because Tor uses a centralized approach to distribute its routers' information, it is possible for censors to obtain the list of the public routers (from the directory servers), and block access to the Tor network. To mitigate this problem, Tor uses special unlisted routers called *bridges* to help users residing within regimes, such as China and Iran, which are actively blocking the Tor network. Clients can obtain information about bridges by visiting `https://bridges.torproject.org` or by sending mail to `bridges@bridges.torproject.org` from a Gmail account. Clients usually get a response consisting of the contact information of three bridges. Clients can then configure their OPs to use the bridges they learn about as first hops in their circuits. To prevent censors from enumerating the list of available bridges, Tor rate limits the distribution of bridges using simple IP- and email-based strategies. For example, users requesting bridges from the same IP/email addresses will get the same list of bridges.

In addition to blocking by IP addresses, censors can also filter by traffic flows. To protect Tor flows from being identified in censored regimes, Tor uses *pluggable transports*, which are extensions that provide bridge users with a means to obfuscate and disguise their Tor traffic to appear like different protocols or applications (such as Skype or HTTP).

3. DESIGN WEAKNESSES

Despite its popularity, previous research has pointed out several design weaknesses in Tor. Below, we summarize those weaknesses.

3.1. Scalability

This refers to the network's ability to grow with the increasing number of users. Scalability problems are lurking in the future of the Tor network because of two main issues: the centralized design, and the high client-to-router ratio.

Recall that the network uses a centralized design to help clients and routers bootstrap and continuously keep a fresh list of routers serving in the network. This is important to prevent some families of partitioning or fingerprinting attacks that leverage stale information to compromise the anonymity of users. For example, if a client is using an old set of routers, whose size is smaller than the actual number of current routers, this will give the attacker a higher success rates in guessing which routers

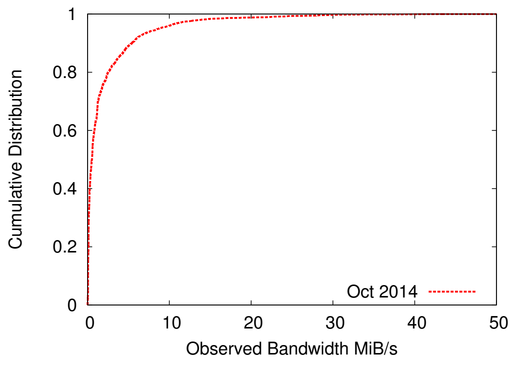

Fig. 4. Distribution of observed bandwidth taken from router descriptors in October 2014. The observed bandwidth is an estimate of the capacity of the router [Dingledine and Mathewson 2015a]. To bootstrap this value, when routers join the network, they build test circuits and send cells over the test circuits in order to measure their bandwidth.

are used in the client's circuits. Therefore, routers and clients download the router descriptors and network consensus document, which lists all current routers and their information, periodically, in order to have a consistent view of the network. This centralized design, while a good defence against some attacks, is a cause for scalability problems. First, it has been estimated that if the Tor network grew to the sizes of popular peer-to-peer (P2P) networks, it would spend more bandwidth relaying descriptors than in routing anonymous traffic [McLachlan et al. 2009].

Second, although the number of Tor users has been steadily growing since it was deployed more than ten years ago, the number of routers supporting the network has not been growing at the same pace. This results in a high client-to-router ratio, where more than one million clients route their traffic over more than 6000 routers.

3.2. Security

Recall that the traditional security analysis of Tor assumes that the attacker needs to control both ends (entry and exit) of a circuit in order to compromise it. However, there have been several proposed attacks that demonstrated their ability to increase the compromise rate of users, such as the Selective Denial of Service (SDoS) attack [Borisov et al. 2007], side-channel information attacks [Murdoch and Danezis 2005; Hopper et al. 2010; Evans et al. 2009; Mittal et al. 2011a], and the Autonomous System (AS)-level adversary attack [Murdoch and Zielinski 2007; Edman and Syverson 2009].

3.3. Lack and Mismanagement of Resources

Because Tor's resources, such as its router bandwidth and CPU, are provided by volunteers, they suffer from significant heterogeneity. For instance, the bandwidth capabilities of routers can vastly range from as little as 20 KiB/s to more than 20 MiB/s. As can be seen in Figure 4, more than 60% of routers have less than 1 MiB/s, and only 10% of the routers have more than 20 MiB/s. Recall that a circuit consists of three routers; the huge difference in capacity results in high variability in performance as observed by the system users, as we saw previously in Figure 3.

This problem becomes even more evident with the increasing usage of bandwidth-constrained routers, such as bridges. It has been observed that circuits that use bridges are significantly slower than those that use non-bridge routers [AlSabah et al. 2013]. Finally, instead of allowing the slowest routers in the network to share some traffic load, Tor's router selection algorithm (described in Section 2.6) favours mostly faster routers since the selection is weighted by bandwidth. Routers that are not assigned the "Fast" flag are not considered as candidates in Tor's router selection algorithm.

3.4. Poor Quality of Service

Traffic congestion adds further delays and variability to the performance of the network. McCoy *et al.*[2008] studied traffic on the Tor network and revealed that although Tor was originally designed for interactive applications such as web browsing and instant messaging, a small number of Tor users use bandwidth-greedy applications such as BitTorrent that consume more than 40% of the available bandwidth in the network; this in turn degrades the experience of interactive application users.

Worse, Tor provides all applications with equivalent service, despite the fact that interactive and bulk applications have different requirements. For example, higher responsiveness is very critical for real-time applications, but is irrelevant for bulk applications, which require higher throughput. Ideally, bulk downloads should be running in the background, so as not to impact interactive applications. If the network has available bandwidth to use for bulk downloads, then such bandwidth-intensive applications can ramp up to use any spare bandwidth.

3.5. Poor Transport Design

Tor's OPs and ORs communicate with each other using TCP connections. Every OR-to-OR TCP connection multiplexes circuits from several users. Reardon and Goldberg [2009] pointed out that this design can potentially hinder the performance of interactive circuits. To understand this problem, consider the application-layer queuing architecture of Tor, depicted in Figure 2. The figure shows the data paths of cells belonging to two circuits built by two different clients, where one circuit is used for file sharing and the other is used for browsing. The clients use two different entry guards, but share the same middle and exit ORs of a circuit. This means that both circuits will be multiplexed in the same TCP connection between the two ORs (middle and exit).

Since the file sharing circuit transmits more data than the interactive browsing circuit, two issues can undermine the performance of the interactive circuit. First, both circuits share an application layer output buffer, which is filled quickly by the file sharing data, forcing the browsing cells to wait behind a long queue. Second, although the bulk circuit drops more data in the connection between middle and exit, TCP's congestion control, triggered due to lost packets, will be applied on both circuits to throttle them, which is not fair for the browsing circuit.

3.6. Lack of Congestion Control

Although the original Tor design claimed to implement a congestion control algorithm, the reality is that the Tor network is actually not congestion controlled, but only flow controlled. Although flow and congestion control are often lumped together as one concept, they implement different functionalities. Flow control is concerned with regulating flow in the network between two endpoints, so that the sender does not overrun the receiver. Congestion control, on the other hand, focuses on techniques that protect the network from congestion, a state in which a network node is overloaded because the rate of its incoming traffic is greater than the rate of its outgoing traffic. In the context of Tor, the implication of the lack of congestion control is that intermediate routers on a circuit cannot react to congestion or reduce the transmission rate of the sender. TCP serves as a good example to distinguish the two concepts. To maintain flow control in TCP, a sender and a receiver negotiate a window size that controls how much unacknowledged data a sender can forward to a receiver. The size of the window matches the transmission rate of the sender with that of the receiver. As for congestion control, TCP implements four different techniques: slow start, congestion avoidance, fast retransmit and fast recovery. In general, those techniques infer the congestion state of the network by utilizing timers and by monitoring acknowledgements in order to adjust the transmission rate according to the inferred conditions.

3.7. Circuit Construction

In the original onion routing design, to construct a circuit, a user creates an *onion* where each layer contains symmetric keys for the corresponding router on the circuit, and information about the next router on the path. One problem with this approach is that it did not provide a *forward secrecy* property, meaning that if a router is compromised, it can reveal information about past user communication. This can be done if an adversary records all communication and later obtains a private key of a router and uses it to decrypt the session key messages, thereby obtaining the key to decrypt the rest of the communication.

To avoid this problem, the circuit construction in Tor is performed in a *telescoping* manner. This means that the circuit is built incrementally and interactively with each hop on the path, as described in Section 2.2. The drawback of this operation is that it is costly; establishing a circuit of length ℓ requires $\Theta(\ell^2)$ network communications, and $\Theta(\ell^2)$ symmetric encryptions/decryptions [Kate et al. 2007].

4. CLASSIFICATION OF IMPROVEMENT TECHNIQUES

Several research directions have spun off to address the design weaknesses described in Section 3. We have visualized those directions, branching off from the center, in the mind map depicted in Figure 5. The leaves of the mind map represent previous proposals that we will cover in this paper. Note that our coverage of the previous proposals will neither be exhaustive nor complete; however, we hope that it will be sufficiently detailed so that the reader will appreciate the core of this research area.

As shown in the figure, those five main research directions can be categorized as follows:

(1) Congestion: Today, research carried out to relieve congestion in anonymity networks is vast. Congestion manifests itself in Tor in two layers: the *overlay* (application) layer (Section 5), and the *transport* layer (Section 6). Research on congestion addresses the poor transport design, the lack and mismanagement of resources, poor QoS, and the lack of congestion control.
(2) Router selection: Recall that an OP (Tor client) has to build circuits before it can use the network. A router selection scheme is concerned with the method by which nodes are selected to form these circuits. Traditional IP routing protocols attempt to minimize cost, which can be a function of various metrics such as hop count, latency, bandwidth, load, etc. While these metrics can help choose the highest-performing paths, they cannot be easily adopted in anonymity networks, where choosing random routers for a path is more desirable from a security perspective. Therefore, there is an increasing need to find routing scheme alternatives that combine the security benefit of choosing random routers to construct a path with the performance benefits of choosing least-costly routers. We see in Section 7 how the router selection problem has been approached in the literature. This research direction also addresses the lack and mismanagement of resources.
(3) Scalability: To address the scalability problems facing anonymity networks, previous work proposed decentralizing the network infrastructure to support peer-to-peer network setups. Because of the anonymity challenges facing P2P networks (P2P lookups break anonymity), another proposed direction is to maintain the centralized architecture of the network, while leveraging Private Information Retrieval

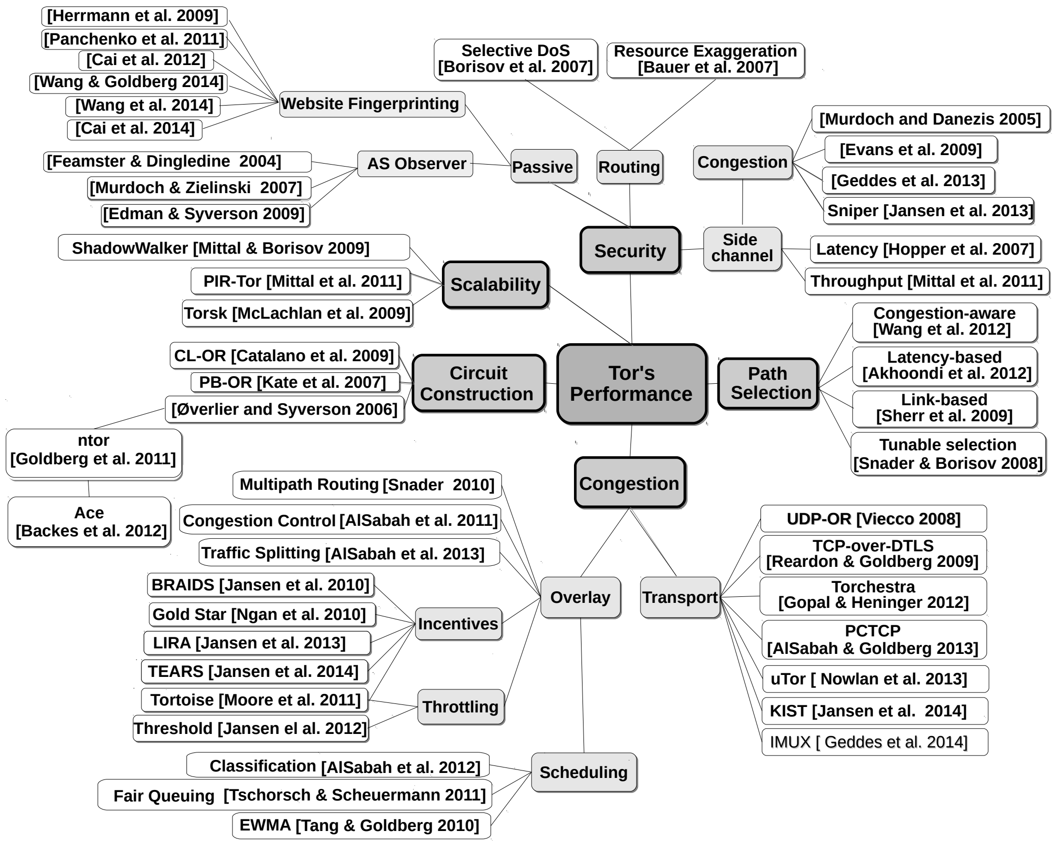
Tor's Performance
Security
Passive
Website Fingerprinting
[Herrmann et al. 2009]
[Panchenko et al. 2011]
[Cai et al. 2012]
[Wang & Goldberg 2014]
[Wang et al. 2014]
[Cai et al. 2014]
AS Observer
[Feamster & Dingledine 2004]
[Murdoch & Zielinski 2007]
[Edman & Syverson 2009]
Routing
Selective DoS [Borisov et al. 2007]
Resource Exaggeration [Bauer et al. 2007]
Side channel
Congestion
[Murdoch and Danezis 2005]
[Evans et al. 2009]
[Geddes et al. 2013]
Sniper [Jansen et al. 2013]
Latency [Hopper et al. 2007]
Throughput [Mittal et al. 2011]
Scalability
ShadowWalker [Mittal & Borisov 2009]
PIR-Tor [Mittal et al. 2011]
Torsk [McLachlan et al. 2009]
Circuit Construction
CL-OR [Catalano et al. 2009]
PB-OR [Kate et al. 2007]
[Øverlier and Syverson 2006]
ntor [Goldberg et al. 2011]
Ace [Backes et al. 2012]
Path Selection
Congestion-aware [Wang et al. 2012]
Latency-based [Akhoondi et al. 2012]
Link-based [Sherr et al. 2009]
Tunable selection [Snader & Borisov 2008]
Congestion
Overlay
Multipath Routing [Snader 2010]
Congestion Control [AlSabah et al. 2011]
Traffic Splitting [AlSabah et al. 2013]
Incentives
BRAIDS [Jansen et al. 2010]
Gold Star [Ngan et al. 2010]
LIRA [Jansen et al. 2013]
TEARS [Jansen et al. 2014]
Throttling
Tortoise [Moore et al. 2011]
Threshold [Jansen el al. 2012]
Scheduling
Classification [AlSabah et al. 2012]
Fair Queuing [Tschorsch & Scheuermann 2011]
EWMA [Tang & Goldberg 2010]
Transport
UDP-OR [Viecco 2008]
TCP-over-DTLS [Reardon & Goldberg 2009]
Torchestra [Gopal & Heninger 2012]
PCTCP [AlSabah & Goldberg 2013]
uTor [Nowlan et al. 2013]
KIST [Jansen et al. 2014]
IMUX [Geddes et al. 2014]

Fig. 5. This mind map summarizes the research directions on Tor. Its leaves represent works that are discussed in this paper.

(PIR) techniques to reduce the costs of router discovery and augment the network's scalability. Section 8 surveys both directions.

(4) Circuit construction: To reduce the computational and communication cost of Tor's iterative circuit building process, several protocols have been proposed. Some protocols trade off between anonymity and performance. In general, performance gains are achieved at the expense of anonymity guarantees when a proposed protocol requires placing more trust in a centralized party, or when forward secrecy is eliminated. Section 9 is dedicated to the circuit construction problem.

(5) Security: Attacks on anonymous communication systems can be categorized to *passive*, where an Autonomous System (AS) adversary can link the source to the destination, *side-channel*, where the adversary leverages congestion, latency or throughput to increase the compromise rates, or *routing-based*, where the adversary exploits the router selection algorithm to attract more client circuits. Those attacks are presented in Section 10.

5. OVERLAY LAYER CONGESTION

In Tor, congestion manifests itself at both the overlay and the transport layers. In this section, we survey techniques that aim to combat congestion at the overlay layer. Those techniques can be categorized into those based on incentives, on traffic prioritization and throttling, on multipath routing, and on congestion control.

5.1. Incentive-based Schemes

Incentive-based schemes aim to encourage clients to contribute bandwidth to the network by becoming ORs themselves. From a high level, all existing proposals—the Gold Star scheme [Ngan et al. 2010], BRAIDS [Jansen et al. 2010], and LIRA [Jansen et al. 2013]—are similar in terms of the incentive used: performance gains.

The Gold Star scheme aims to incentivize Tor clients to relay anonymous traffic. Trusted authoritative directories are responsible for assigning "gold stars" to routers after evaluating their bandwidth capabilities, and after testing their faithfulness in relaying traffic. A gold star's router traffic is given higher priority by other routers, which means they always get relayed ahead of other traffic. In this scheme, there are only two types of traffic: gold star prioritized, and other non-prioritized. The reason for the simple two-level classification of the proposed scheme is to protect the anonymity of users that can be hurt if an observer is able to reduce the anonymity set based on the class of service of a circuit. This scheme is very simple and easy to implement in the current Tor network. Experimental results have shown that cooperative users—clients that donate an amount of bandwidth to relay network traffic—witness significant improvements in download times and ping times, even under heavy background traffic. Also, cheating users that stop relaying network traffic after achieving a gold star status are penalized with degraded performance whenever they toggle to their selfish behavior.

In BRAIDS, clients first obtain free *tickets* from a bank, which is a centralized and partially trusted offline entity that is responsible for bandwidth accounting tasks. Clients remain anonymous, as they use blind signatures to get the signed tickets from the bank. Each ticket can be redeemed at a specified router; this solves the problem of double spending, meaning that no client can use a ticket twice. When a client wishes to get improved service from a router R, it can present it with tickets. Each ticket allows the client to receive an improved prioritized service for a fixed number of data cells. After the ticket is used, the circuit priority is lowered, but can be restored after a new ticket is presented. A used ticket is utilized by R as a voucher that is redeemable for a new router-specific ticket for another router C. Therefore, the more service a router R provides, the more tickets for improved service it can collect, thereby obtaining a better service from the network.

While experimental results showed that BRAIDS leads to better overall network performance (assuming more clients provide router service), it is still impractical as it relies on a centralized server for distributing and exchanging tickets (for millions of users) resulting in scalability and single-point-of-failure problems. Furthermore, both the Gold Star scheme and BRAIDS introduce anonymity risks as they allow an observer to infer if the client is also a router (based on service obtained), and thereby reduce the size of the anonymity set from more than a million users to only around 6000 routers.

LIRA was proposed to address some of these limitations by leveraging a cryptographic lottery at routers. If a client wants a prioritized service, he makes a guess and gets prioritized service with a tunable probability. To get guaranteed prioritized service, clients can run as routers and in exchange for their bandwidth, they obtain tokens from a central bank. Clients can then use those tokens to purchase "guaranteed winners" to the lottery for prioritized service at the routers. The main drawback of LIRA is that misbehaving clients can try to create multiple circuits until they get a prioritized one. For example, if the probability of winning the lottery to obtain improved service (in the absence of a "guaranteed winner" ticket) is 20%, then on average, clients will get 2 prioritized circuits out of every 10 they build; this will encourage clients to keep building circuits to find winners, wasting network resources. Reducing the probability of winning introduces anonymity issues, because an adversary monitoring a prioritized circuit can reduce the anonymity set. On the other hand, LIRA addresses BRAIDS' scalability issues because LIRA's bank only interacts with ORs, and not with all clients.

TEARS [Jansen et al. 2014a] attempts to solve the scalability problems by distributing the management of incentives over multiple semi-trusted parties, such as the authoritative directory servers already employed by Tor. To prevent the semi-trusted servers from misbehaving, their interactions with routers use an e-cash protocol (similar to Bitcoin), which is publicly auditable. Routers are rewarded, by the servers, with anonymous coins (called Shallots) in exchange for the bandwidth they spend relaying anonymous traffic. Shallots can be exchanged for PriorityPasses, which are router-specific tickets that can be used to obtain prioritized network service. The trusted servers also distribute Shallots to users in order to increase the anonymity set of routers. One issue challenging TEARS is that it relies on a decentralized process to perform secure bandwidth measurements, which does not exist in practice in the current Tor network. While bandwidth authorities are currently deployed by the Tor network, it is not clear how reliable their measurements are in the face of various adversarial scenarios.

5.2. Scheduling and Traffic Prioritization

5.2.1. Circuit Scheduling. To address the cross-circuit interference problem (described in Section 3.5), Tang and Goldberg [2010] propose prioritizing circuits that have interactive traffic. Because it is hard to identify the type of traffic in an anonymity network, the insight of this work is to change the scheduling algorithm so that it gives precedence to circuits that have sent fewer cells recently, thereby increasing their responsiveness. To achieve that, each circuit maintains a state variable that keeps track of the exponentially weighted moving average (EWMA) of the number of cells sent. A smaller such value is likely to identify a circuit with interactive traffic, which the scheduler prioritizes and gives service next, as opposed to Tor's original round-robin scheduling. Small-scale live experiments show that this technique does not hurt the performance of bulk transfer circuits; however, later work by Jansen *et al.*[2012] showed that the performance of bulk clients can be noticeably worse when this prioritization algorithm is used. Also, web clients only benefit from the EWMA circuit scheduler under heavy traffic loads. Under light traffic loads, clients might experience performance degradation when EWMA is used.

5.2.2. Fair Queuing. Tschorsch and Scheuermann [2011] observe that the interaction between the router-level token bucket rate-limiting algorithm and the three levels of scheduling (incoming and outgoing round robin scheduling at the connection level, and outgoing scheduling at the circuit-level) results in an unfair allocation of bandwidth between the competing circuits. The router-level throttling algorithm divides the bandwidth equally among connections, even though some connections carry more circuits. The proposed solution is to allocate any unutilized bandwidth from one connection to other connections following max-min fairness principles. This can be achieved by using a global round robin scheduler that serves all circuits. This technique is used in combination with N23 (described in Section 5.4) in order to get quick feedback on congestion at bottlenecks. Simulation results showed that using this approach results in global max-min fairness allowing a fair bandwidth allocation among circuits.

5.2.3. Classification. One problem in Tor is that it provides all applications with equivalent service, despite the fact that different applications have different requirements. For example, interactive real-time applications require more responsiveness from the network, whereas bulk applications require higher throughput. To alleviate the impact that bulk circuits have on interactive circuits, AlSabah *et al.*[2012] introduced DiffTor. DiffTor is a machine-learning-based approach to perform traffic classification of Tor's encrypted traffic into three traffic categories: browsing, streaming, and bulk. This approach has two use cases: *online* and *offline*.

The offline classification can be used to provide insight into the usage of the network without applying privacy-intrusive Deep Packet Inspection (DPI) techniques. The online classification, on the other hand, gives each router the flexibility of applying *real-time* Quality of Service rules on the different traffic classes. For example, a router can prioritize interactive circuits, and throttle bulk circuits if it is experiencing congestion. When the router has some spare bandwidth, it can allow the bulk circuits to ramp up.

Since the traffic is encrypted and travels in fixed-sized units, the authors rely on timing and cell counts (and other related statistics) to extract features for the classification. More than 90% accuracy can be obtained using decision trees and Naive Bayes classifiers. To evaluate the effectiveness of the classification approach, they implemented a Naive Bayes classifier on an entry guard router that they deployed on the live Tor network. They also implemented a simple QoS rule that throttles a connection if several circuits in the connection are classified as bulk. This simple rule resulted in more than 70% improvements in the time-to-first-byte and download time measurements observed by a web client using the same entry guard. This demonstrates how greedy applications are capable of consuming most of the bandwidth in Tor if left uncontrolled.

5.2.4. Throttling. To reduce the impact of greedy file sharing applications on the network, Jansen *et al.* [2012] leverage Tor's already existing token bucket throttling system (Section 2.5). This technique is employed by entry guards, who keep track of the EWMA of cells sent (as proposed previously by Tang and Goldberg; see Section 5.2.1) at the *connection level*. Furthermore, they explore three algorithms that adjust or throttle the connection between an entry guard and a client. Their most effective algorithm, nicknamed the threshold algorithm, sorts circuits from loudest to quietest based on the EWMA of cells sent, with the goal of throttling a loudest threshold of all connections. The authors report that a threshold of 90% yields the best results. The problem with such an approach is that it would unnecessarily throttle time-sensitive interactive applications if the threshold is too high. Another problem with the threshold algorithm is that it is based only on a simple metric (EWMA of cells sent), which is sometimes unreliable as interactive circuits sometimes have large values of EWMA while they are downloading web pages. This metric is also easy to game, as misbehaving bulk clients can try to spread their traffic over multiple circuits so that their circuits look lighter [AlSabah et al. 2012].

5.2.5. Hybrid Techniques. In Tortoise [Moore et al. 2011], all clients using the network are throttled, using Tor's already-implemented token-bucket system (Section 2.5), to a certain bandwidth rate by entry guards. While the global rate-limiting significantly slows down bulk downloaders, and allows more spare bandwidth to be present in the network, it also slightly slows down web browsing users. For a user to bypass the throttling, he has to run as a router. This design unfortunately hinders the anonymity of router operators, as if the adversary observes an unthrottled circuit, he can reduce the anonymity set of the unthrottled circuit initiators to the set of publicly known router operators.

To prevent users from gaming Tortoise by either running limited-bandwidth routers, or by running only sporadically, global throttling can only be bypassed by the routers labeled "fast" and "stable" by the directory authorities. This condition ensures that clients can only bypass the throttling if they contribute a significant amount of bandwidth to the network.

5.3. Multipath Routing

Multipath routing has been studied in the context of onion routing by Snader [2010]. In this work, Snader simulated downloading a 1 MB file over a Tor network simulator. The file was divided into 512-byte chunks and sent over multiple circuits simultaneously. He observed that throughput was significantly improved with the use of multiple circuits; however, using two circuits performs better than using one circuit or more than two circuits—using more than two circuits increases the chances of choosing a slow router. The median transfer time remains unchanged for more than two circuits, but the 90^{th} percentile transfer times actually increase when the number of circuits used is greater than two. However, this work does not provide a security analysis of how compromise rates would be affected if clients adopted this technique.

Multipath routing has also been proposed to improve the experience of bridge users. Because bridge users observe worse performance than the rest of the network, Conflux [AlSabah et al. 2013] was introduced to enhance their browsing experience by using multipath routing in Tor. A bridge user starts by building a *primary* circuit using Tor's usual weighted bandwidth selection algorithm. Next, he builds a *secondary* circuit using the same algorithm with the exception that it has the same exit OR as the primary circuit, and uses a different bridge. Once the two circuits are built, the user sends a new command message that contains a special 32-bit number on both circuits towards the common exit. This allows the exit to know that both circuits belong to the same user. When the user starts downloading data, the exit performs a weighted traffic splitting algorithm that balances load on each circuit. Conflux exploits Tor's circuit_sendme messages sent from the client to the exit on each circuit to compute the latency on each path, and set the weight of the traffic sent on each circuit accordingly. If one path is more loaded, it gets a smaller fraction of the traffic. Because cells may arrive out of order at the client, a Conflux exit adds 4-byte sequence numbers to the cells, and a Conflux client buffers and reorders cells before passing them to the client.

The location of the bottleneck node in the circuit affects the benefits observed by Conflux. Benefits appear to be substantial as long as the Conflux circuit bottleneck is not the (common) exit. Furthermore, because Conflux uses more nodes in its circuit than a vanilla Tor circuit, it slightly raises the path compromise rate for its users.

5.4. Congestion Control

Although traffic congestion is a major problem facing any network, previous work [AlSabah et al. 2011] pointed out that Tor is flow controlled, but not congestion controlled. This means that intermediate routers on a circuit cannot react or protect themselves from congestion at the overlay layer. To introduce congestion control in Tor, AlSabah *et al.*[2011] introduced to Tor `N23`, a credit-based congestion control algorithm that was originally proposed for Asynchronous Transfer Mode (ATM) networks. Briefly, every OR maintains a variable, `credit_balance`, which denotes the amount of data an OR can send on a circuit. When a circuit is created, the `credit_balance` is initialized to `N2+N3`, where `N2` and `N3` are system parameters. When an OR forwards a cell downstream, it decrements the circuit's balance by 1. Every time an OR sends `N2` cells downstream, it forwards a new cell type, known as the credit cell, upstream. The credit cell allows the upstream OR to calculate the amount of buffer space available downstream to update its `credit_balance` variable accordingly.

`N23` provides several advantages to Tor ORs. First, their circuit queues depicted in Figure 2 never grow beyond `N2+N3` (approximately 100), whereas in stock Tor, the circuit queues can reach up to 1000 cells. Second, when an OR gets congested, it stops sending credit cells to the upstream OR, which will run out of credits, and stop forwarding data to the congested OR. This allows ORs to signal congestion using back pressure. Finally, if there is a sudden increase or decrease in the available bandwidth, `N23` reacts within the round-trip time (RTT) of a single link, as opposed to the end-to-end RTT of Tor's window-based flow control algorithms.

6. TRANSPORT LAYER CONGESTION

Recall that one of the key culprits to Tor's poor performance is its transport design. Tor multiplexes several circuits over the same TCP connection, which results in unfair application of the TCP congestion control of the shared connection on all circuits.

6.1. TCP-over-DTLS

Another implication of the cross-circuit interference problem that was observed by Reardon and Goldberg is that when packets are dropped or reordered from one circuit, all other circuits going through the same TCP connection are penalized. The reason is that the TCP stack will buffer available data on input buffers until the missing in-order component is available. To solve the cross-circuit interference problem, Reardon and Goldberg proposed TCP-over-DTLS, a transport design in which every circuit is managed by a user-level TCP connection. DTLS, a secure datagram protocol, is used for inter-OR communication. This design maintains hop-by-hop reliability and congestion control, but it introduces performance and deployment issues: (1) there is a lack of an available user-level TCP stack whose license is compatible with Tor, (2) even if such user-level stacks exist, they are known to perform worse and consume more CPU cycles than their kernel-level counterparts [Edwards and Muir 1995; Braun et al. 1995].

6.2. UDP-OR

In UDP-OR [Viecco 2008], a TCP connection is maintained at the two ends of a circuit, the OP and the exit, and UDP, an unreliable transport, is used for communication between routers. The advantage of this design is that it simplifies the tasks of processing at intermediate routers; however, it creates the following problems: (1) reliability and in-order delivery is not maintained between routers, which requires changing the current cryptographic protocols currently used in Tor in which if a cell is lost, subsequent cells can not be encrypted or decrypted correctly. This also means that discovering lost packets, in cases of extreme congestion, will be in the order of several seconds; (2) this design used the kernel TCP stack, which opens the door to OS fingerprinting attacks.

6.3. Torchestra

In Torchestra [Gopal and Heninger 2012], two TCP connections are used between any pair of routers. One connection is used to multiplex light circuits, and the other is used to multiplex heavy circuits. The exit OR uses the EWMA of cells sent (as proposed previously by Tang and Goldberg; see Section 5.2.1) in order to classify circuits as light or bulk. Then, special control cells are sent to instruct the remaining routers on the circuit of the classification decision. Therefore, all ORs in a circuit using Torchestra need to upgrade in order to benefit from this work. However, it has been pointed out that the EWMA of cells sent is not enough to determine if a circuit is light or bulk, and unfortunately, the classification accuracy was not discussed in Torchestra. Although

Torchestra was implemented, it was only tested on a small-scale network setup, which is not sufficient to provide conclusive results regarding its benefits.

6.4. PCTCP

AlSabah and Goldberg [2013] attempt to solve the shortcomings of previous transport designs by dedicating a separate kernel-level TCP connection for each circuit. This has the advantage of maintaining reliability and congestion control on a hop-by-hop basis. Second, using a separate kernel-level TCP connection eliminates buffer sharing between bulk and interactive application cells, which solves the cross-circuit interference problem both at the overlay and the transport layers. Finally, this design eliminates the performance and deployability problems inherent from using a user-level TCP.

However, to prevent an adversary to determine which circuit within a connection is carrying any particular cell, PCTCP proposes using the Encapsulation Security Payload (ESP) protocol of IPsec, the standard security mechanism for the IP layer, in transport mode, which provides authenticity, integrity and confidentiality of the data communicated between any two ORs using PCTCP. The individual circuits, each in their own TCP connection, are all bundled within a single IPsec channel, so the attacker cannot discern the circuits.

Although the experimental evaluation of PCTCP shows significant improvement benefits, deployment of IPsec might face some challenges in Tor, which could be mitigated by shipping Tor with scripts that can facilitate router operators installing IPsec. Other challenges might arise from older Windows-based ORs, which are notorious for their inability to handle the increasing number of sockets required for the operation of PCTCP.

6.5. uTor

Nowlan *et al.*[2013] introduced uTor to solve the *head-of-line blocking* issue, where if one heavy circuit, occupying the first packet in the TCP congestion window, drops data, the TCP congestion window will buffer available data from other circuits and will not advance the TCP window until the missing packet is available. This is done to guarantee in-order delivery for the data in one TCP connection.

In Tor, since circuits are logically independent from each other, delaying data from one circuit until all missing data is available from other circuits is not needed. uTor solves this problem by using Unordered TCP (uTCP) protected by Unordered TLS (uTLS), which allows the TCP connection to send the available data even in the condition of missing dropped data. This way, a lost packet from one circuit will not interfere with other circuits' delivery. uTor requires very minimal changes to the Tor source code, and the upgrade is backward compatible. However, the micro-benchmark evaluation shows that only modest performance gains can be obtained using uTor.

6.6. KIST

Jansen *et al.*[2014] confirm previous findings [Reardon and Goldberg 2009] regarding where cells spend most of their queuing time in their data path in a circuit. Both Jansen and Reardon agree that cells spend most of their time waiting in the socket output buffers waiting to be flushed. In addition, Jansen *et al.*observe that although Tor uses its EWMA circuit scheduler (see Section 5.2.1) to prioritize cells from light circuits, those cells lose their priority when they are written to the kernel socket. The reason is that they are not written to the kernel according to their priority level, but according to when their corresponding sockets receive notifications from libevent, the asynchronous event library that Tor uses for network reading and writing operations.

To solve this problem, Jansen *et al.*propose KIST (Kernel Informed Socket Transport), which ensures that cells do not lose their priority at the kernel. First, when a

KIST-upgraded OR is given a ready socket by libevent, instead of writing cells right away, it waits until the end of a tunable time interval, during which it keeps track of all ready sockets reported by libevent. At the end of the interval, it pulls cells from the highest-priority circuits and writes them to their corresponding sockets. That way, a higher-priority circuit is written first to the kernel and gets flushed first to its next destination since the kernel services sockets in order. Note that this achieves global circuit scheduling, as opposed to Tor's per-connection local circuit scheduling.

Furthermore, KIST measures the amount of data that can be sent to the socket based on the minimum of the available outbound kernel buffer space, and the available congestion window space. While this moves the queueing delays from the kernel to Tor, it gives Tor the ability to control the prioritization until the last instant before kernel starvation. Both components of KIST have been implemented in Tor and large-scale experiments showed significant performance benefits. Like many performance enhancement proposals, however, decreasing network latency makes the network more vulnerable to latency-based attacks, where the adversary uses latency as a side channel to deanonymize clients (see Section 10.3).

6.7. IMUX

Geddes *et al.*[2014] observe that socket exhaustion attacks are effective against PCTCP, Torchestra, and vanilla Tor, as an adversary can use one or multiple clients to anonymously create a large number of circuits through a target router. This is done to force the router to reach its maximum connection limit `Connlimit`, which is set by returning the maximum allowed open file descriptors from the OS. Note that the attack is most effective when ORs keep the default connection limit value of 4096 for Linux and BSD OSs, and 15,000 for other OSs. When an OR reaches its limit, it denies connections and becomes unusable by other clients. If an OR operator allowed unlimited connections, the attack may still be effective because opening circuits can exhaust other resources of the OR, such as memory, CPU, and bandwidth.

IMUX solves this problem by introducing a connection manager and scheduler. For connection management, a function is called periodically (every second) to calculate an upper bound of the number of connections allowed in a specific channel (logical connection between two routers). First, a global connection limit is calculated by multiplying `ConnLimit` by τ (a tuneable parameter). This global limit is then multiplied by the fraction of active circuits in the channel in question. After that upper bound is calculated, the connection manager decides if it can open (or close) separate connections, when new circuits are created, to get closer to the bound calculated. If it needs to close connections, it will choose connections that have not been fully opened, followed by connections that were created most recently, and finally, followed by the ones that were least recently used.

Recall that in PCTCP, the mapping is one to one between circuits to connections. However, since IMUX caps the number of connections allowed, and since the total number of circuits might exceed the number of connections, there is a need to manage the mapping between circuits and connections. This is done using the connection scheduler. Geddes *et al.*experiment with three scheduling schemes: round robin, EWMA-based mapping, and shortest queue scheduling.

Round robin scheduling is similar to PCTCP in the sense that every circuit will get a separate connection; however, if there are more circuits than connections, circuits will be multiplexed and assigned to connections in a round robin manner. The EWMA mapping scheduling generalizes Torchestra's technique to more than 2 connections; that is, EWMA values are maintained for circuits and connections, and are used to sort circuits and connections from the least to the most noisy. Then, circuits are mapped to connections that have similar EWMA values. Finally, in shortest queue scheduling, circuits are assigned to the least congested connections based on their queue sizes.

Experimental evaluation demonstrated that for web clients, the round robin scheduling approach outperformed the other two approaches. The problem with the shortest queue scheduling approach is that light circuits might get multiplexed with loud circuits, thereby affecting their performance. EWMA mapping scheduling does not provide any improvement over stock Tor because the EWMA value of light circuits can experience sudden spikes when bursts occur, which takes time to decrease. This results in subsequent delays for those circuits.

7. ROUTER SELECTION

Recall that Tor clients select routers for circuits in a frequency roughly proportional to their bandwidth. This section surveys previous research papers that investigated the router selection problem in Tor and aimed to improve it by trading off its performance with anonymity, by exploring other link-based metrics such as latency and jitter, by exploiting the geographical location of routers to reduce latency, or by enabling clients to be congestion aware.

7.1. Tunable Selection

Snader and Borisov [2008] identified two key weaknesses in Tor's original design. First, ORs' bandwidths are self-reported. This allows some routers to misbehave by reporting exaggerated bandwidth capabilities in order to lure more circuits to use them [Bauer et al. 2007], in order to increase the rate of compromise. Even when honest bandwidth values are reported, they are still a poor indicator of the available capacity because of network dynamics and congestion states. Second, the Tor design does not provide users with the flexibility to trade off anonymity with performance according to their requirements.

Therefore, to solve the self-reported bandwidth problem, they proposed an *opportunistic bandwidth monitoring* approach, where every router aggregates the bandwidth capabilities of other routers it contacts over time, and then reports these measurements to the bandwidth authorities. They also introduce *Tunable Tor*, an algorithm that allows users to configure the level of performance they want to trade off with anonymity. Briefly, Tunable Tor works as follows: the list of ORs is sorted according to some criterion (such as the opportunistic bandwidth measurement). If this list is indexed from 0 to $n-1$, then the router selected is that with the index $\lfloor n \cdot f_s(x) \rfloor$, where x is selected uniformly at random from $[0,1)$, and f_s is a family of functions $f_s : [0,1) \to [0,1)$ given by $f_s(x) = \begin{cases} \frac{2^{sx}-1}{2^s-1}, & s \neq 0 \\ x, & s = 0 \end{cases}$.

Configuring a higher value for s results in a selection prejudice towards routers with higher ranking in the list. If $s = 0$, the router is chosen uniformly at random.

Murdoch and Watson [2008] compared the performance of four different Tor path selection algorithms: the original uniform router selection, Tor's current bandwidth-weighted router selection, and Tunable Tor with the minimum suggested s for improved anonymity ($s = 1$), and the maximum suggested s for improved performance ($s = 15$). In their evaluations, they used two performance metrics: probability of path compromise and network latency. They used queuing theory to model the latency expected with the different path selection algorithms.

Their latency results demonstrated that Tor's weighted bandwidth selection algorithm provides improved performance over the other router selection algorithms. The tuneable algorithm by Snader and Borisov was also found to degrade performance, as if all clients choose to get higher performance, the fastest routers will be congested and overloaded. Moreover, Tor's weighted bandwidth selection algorithm also showed improved anonymity against a node-rich and bandwidth-poor attacker. The reason is that when higher-bandwidth nodes have a higher probability of being selected, the algorithm deviates further from selecting malicious poor-bandwidth nodes. If the attacker introduces a handful of high-bandwidth nodes, the authors found that Tuneable Tor with $s = 15$ and the bandwidth-weighted algorithm yielded the worst compromise rate compared to Tuneable Tor with $s = 1$, and the uniformly random selection of routers.

7.2. Link-based Selection

Sherr *et al.*[2009] note a number of problems in the opportunistic bandwidth monitoring described above. Routers reporting the bandwidth of other monitored routers can lead to two undesirable effects. First, routers can lie about the bandwidth of other routers they report. Colluding malicious routers can report exaggerated bandwidth capabilities about members of their coalition. This problem, however can be addressed using Eigenspeed [Snader 2010], which is Snader's improved opportunistic bandwidth measurement algorithm that is resilient in the face of malicious attacks. The second problem that Sherr *et al.*point out is that for every router to report the performance of other routers contacted means revealing information about established circuits, giving the servers a more powerful global view of the network.

Alternatively, they propose replacing the opportunistic measured bandwidth in Tunable Tor with a *link-based* metric. Their observation is that choosing paths based on link characteristics such as latency, jitter, or number of traversed Autonomous Systems can provide improved performance over node-based characteristics alone. Their proposed link-based Tunable Tor takes place in two phases. In the first phase, the initiator generates various candidate paths, and then the end-to-end cost of each path is computed according to the desired link-based metric. In the second phase, paths are sorted according to their metric, and the Tunable Tor algorithm is used to trade off between performance and anonymity.

One assumption made in the link-based router selection algorithm is that the initiator must maintain knowledge of the costs of the whole network, in order to be able to compute the cost of the whole path. For example, if a user wishes to use latency as a metric in constructing circuits, then it must measure the pairwise latency between every two routers in the network. The cost of this measurement can outweigh the benefits of exploiting link-based metrics. For that reason, Sherr *et al.*also propose the use of a *network coordinate system*, a multi-dimensional space in which the distance between routers in the virtual coordinate corresponds to the metric utilized in the router selection algorithm.

7.3. LASTor

Akhoondi *et al.*[2012] proposed a client-side router selection algorithm called LASTor, which exploits the geographical location of Tor routers in order to minimize latencies observed by clients. The router selection approach uses a tunable weighted shortest path algorithm that allows clients to trade off anonymity and performance. LASTor also protects clients from observers at the AS level, as it implements a lightweight technique to reliably avoid creating paths that have the same AS on the path between the client and its entry guard and the path between the exit and the client's destination. Later research by Wacek *et al.*[2013] (Section 7.5), however, demonstrated that LASTor performs poorly when tested under a more realistic network setup.

7.4. Congestion-aware Router Selection at Clients

Wang *et al.*[2012] proposed a path selection algorithm that uses latency as an indicator for circuit congestion. First, Tor's default bandwidth-weighted router selection algorithm is maintained to build circuits. Then, the proposed algorithm uses opportunistic and active-probing techniques to obtain latency measurements for circuits. The client remembers the latency measurements for the individual routers, which can be useful in two ways. First, if a client is using a congested circuit, it can switch to a less-congested circuit. Second, the router selection algorithm is also modified to take into account the latency measurements, in addition to the bandwidth, for candidate routers chosen to build circuits.

7.5. Comprehensive Evaluation

Wacek *et al.*[2013] evaluate the performance and security of all router selection algorithms described above. The authors use an emulated scaled-down Tor network that realistically models the live Tor network. The authors then implement the router selection algorithms and compare their performance in terms of throughput, time-to-first-byte and average ping times. They also evaluate the anonymity of these algorithms using the Gini coefficient, entropy, and compromise rate by an AS-level adversary. The evaluation shows that the congestion-aware algorithm proposed by Wang *et al.*outperforms other router selection algorithms without reducing anonymity. LAS-Tor, on the other hand, provided the poorest performance among other algorithms, but maintained high anonymity guarantees.

8. SCALABILITY

Improving the scalability of the Tor network has taken two approaches. The first approach is to replace the centralized design with a peer-to-peer design. The second direction is to maintain the centralized design, but use PIR during circuit construction. Below we survey those directions.

8.1. Peer-to-peer Approaches

8.1.1. ShadowWalker. ShadowWalker [Mittal and Borisov 2009] was designed with the goal of alleviating the scalability and vulnerability issues of earlier anonymity systems like Morphmix [Rennhard and Plattner 2002] and Tarzan [Freedman and Morris 2002]. It uses DHTs as a foundation to build a redundant structured topology. In addition to fingers (DHT neighbors), each node maintains a list of shadow nodes that can be verified by a simple mathematical formula. The main purpose of shadows is to prevent *route capture* attacks, where a malicious node mis-answers lookup queries, giving only other cooperating nodes as replies. In ShadowWalker, every node maintains signatures, which can be viewed by requesters, from its shadows to certify the correctness of the information in its routing table. An initiator I constructs circuits by contacting a random finger A and asking it to return the finger B with random index i from its routing table. A then returns the contact info of B and the shadow signatures of A, which the initiator can use to check the validity of B. The same process is repeated to extend the circuit. Schuchard *et al.*[2010] showed that ShadowWalker is susceptible to the *eclipse* attack, where malicious nodes attempt to corrupt the routing table of the victim by filling it with addresses of colluding nodes in the P2P network in order to intermediate all its communication, and the *selective denial of service* attacks, where the adversary hinders the reliability of the circuit construction in hopes of increasing the end-to-end compromise rate in anonymity networks (more details in Section 10.2).

8.1.2. Torsk. To solve scalability problems in Tor, Torsk [McLachlan et al. 2009] proposes a *decentralized circuit construction* scheme that uses a combination of a DHT structure for bootstrapping, and a *buddy selection protocol* for peer discovery. To bootstrap, Tor's directory authorities in Torsk are given the role of the *Neighbourhood Authority* (NA), an entity responsible for issuing certificates to neighbors in the DHT space when nodes join or leave the network. This allows bootstrapping to be a low-cost operation, as new nodes are only required to generate a new ID and perform a lookup on their ID to find the closest neighbor, as opposed to downloading the whole topology. Next, the newly joining node contacts the NA, which takes the ID of the node and the certificate of its closest neighbor to generate a new certificate for all affected neighbors.

Router selection is carried out as follows. First, every router uses the buddy selection protocol to find other random routers to consult during DHT lookups. Likewise, a client uses the buddy selection protocol to begin a random walk in the network to discover an entry guard. Next, for subsequent routers, the client uses the last OR on the partially constructed circuit R_i to randomly find R_{i+1} by consulting R_i's previously found lookup buddies. Finally, because lookups are loud (queries are broadcast to many entities in the P2P network) and can reveal information about the constructed circuit, cover traffic is also used to add noise in case lookups are profiled.

Despite this defence, it has been shown subsequently by Wang *et al.*[2010] that an adversary with 20% compromised nodes can compromise 80% of the circuits. The reason is that DHT lookups are loud and they allow an adversary to link what routers a client is searching for.

8.2. A Scalable Centralized Approach: PIR-Tor

A major problem with the two above proposals is that the research community does not currently have a solution for looking up information in a P2P network without revealing the lookup to many parties. However, private information retrieval (PIR) allows a client to look up information in a client-server model without revealing any information about what was being requested, even to the server itself. Therefore, PIR-Tor [Mittal et al. 2011b] steps away from the P2P paradigm to address the scalability problem in Tor, instead advocating for the use of PIR techniques in order for clients to be able to download a small portion of the network view without hindering their anonymity by revealing *which* portion. The current approach in Tor is for clients and routers to regularly download the *entire* network state and then choose their desired records, a costly operation and a significant contributor to Tor's scalability problems. The advantages of PIR-Tor over P2P designs is twofold. First, the client-server bootstrapping architecture of Tor is preserved, making for an easier deployment path. Second, the security guarantees of the system are easier to analyze than the above P2P designs.

The authors investigated two flavors of PIR techniques: computational PIR (CPIR) and information-theoretic PIR (IT-PIR). For CPIR, some routers are selected to act as the PIR servers. To build a circuit, a client contacts one of the CPIR servers to perform two PIR lookups: one for a middle router and another for an exit router. On the other hand, since IT-PIR requires the use of multiple servers (the privacy of a user's query is guaranteed if a threshold number of the IT-PIR servers do not collude), IT-PIR server functionality can be implemented between a client and its entry guards. This reduces the PIR lookup for each circuit to only one to lookup an exit node, while a middle node can be retrieved by a normal lookup. Evaluation of PIR-Tor showed that both techniques of PIR help reduce the communication overhead as the network scales. However, only IT-PIR provides the same level of security of the current Tor network, as the computational overhead of CPIR requires that clients reuse the retrieved routers to build multiple circuits.

9. CIRCUIT CONSTRUCTION

We next survey several proposals that aim to improve the communication and computation overhead of Tor's circuit construction algorithm.

9.1. Improved Diffie-Hellman Based Key Agreement

Øverlier and Syverson [2007] introduced four protocols that aim to reduce the communication and computation overhead of circuit construction in Tor. In their *first* protocol (which is the basis for all their subsequent protocols), RSA encryption, which was used for circuit construction prior to the current ntor protocol (described below), is replaced with a DH key agreement in order to reduce the computational cost.

The *second* protocol uses the first protocol and creates a circuit by sending one command cell to build a circuit in one pass, which reduces communication costs at the expense of forward secrecy. The insight of the *third* protocol is that since the link between the client and the first router is TLS encrypted, there is no need to use a DH key agreement, but they can simply exchange symmetric keys.

Finally, the *fourth* protocol proposes a new key agreement protocol using the ephemeral keys of both the client and router and the long-term keys of the router. The router R_B publishes a long term key g^b. When a client wants to establish a session key with R_B, it computes and sends an ephemeral public key g^x to the server. Then, the server computes the shared key $(g^x)^{b+y}$, and sends g^y to the client, which computes $(g^b g^y)^x$.

Goldberg *et al.*[2011] demonstrate a Man-In-The-Middle attack against this fourth protocol, and propose a fix they call ntor. In ntor, the shared key is computed as $H((g^x)^y, (g^x)^b)$ instead of $(g^x)^{b+y}$ as above. Ace [Backes et al. 2012] was subsequently proposed to slightly improve on the computational cost of ntor. The client sends (g^{x_1}, g^{x_2}) as an ephemeral key to the router, which responds with g^y. The client and router then compute the session key as $(g^b)^{x_1}(g^y)^{x_2}$ and $(g^{x_1})^b(g^{x_2})^y$, respectively. The ntor protocol is currently employed in the Tor network.

9.2. Pairing-Based Onion Routing

Kate *et al.*[2007] propose replacing the circuit construction scheme in Tor with a pairing-based onion routing (PB-OR) protocol that uses a pairing-based non-interactive key agreement protocol. In order for their scheme to achieve unilateral anonymity (meaning that the client authenticates a router without leaking the client's identity), they use an identity-based infrastructure. A trusted entity known as the *private key generator* (PKG) takes a router's well-known identity ID, and uses a master key only known to the PKG to generate a private key d for the router. Clients, on the other hand, can independently generate as many pseudonyms as they need, along with the private keys corresponding to each pseudonym. Then, the client achieves anonymity during the key agreement phase with routers by presenting a different pseudonym with each router; routers use their private keys d to complete the key agreement. Because the key agreement protocol is non-interactive, it significantly reduces the communication overhead of the circuit construction compared to Tor, and it allows a circuit to be constructed in one pass. However, the PKG is able to decrypt all messages encrypted for clients, a single-point-of-failure-problem. Also, to maintain forward secrecy, routers are required perform costly communications with the PKG in order to change their identity keys frequently.

9.3. Certificateless Onion Routing

Catalano *et al.*[2009] note the problems inherited from the use of a PKG in the PB-OR scheme described above and propose to improve it by replacing the anonymous pairing-based key agreement with an anonymous certificateless key agreement scheme. The idea of this scheme is that a client obtains *partial secret keys* from the trusted entity (key generation center KGC), from which he can compute public/secret key pairs PK and SK. Therefore, the newly computed private key SK is not known even to the KGC, and the pair PK/SK can be used to generate several pseudonyms as needed. The rest of the protocol is very similar to that of PB-OR. Another advantage with this approach is that routers can update their keys locally without contacting the KGC.

10. ATTACKS ON ANONYMITY

We next survey various attacks on Tor that aim to hinder the anonymity it provides to users. We examine three categories of attacks: passive attacks, active attacks against clients' choice of routers, and side-channel attacks based on throughput, latency, or congestion.

10.1. Passive Attacks

10.1.1. AS-Level Adversary. Edman and Syverson [2009] argue that the security guarantees provided by Tor are not as originally thought, especially in the face of an AS-level adversary. An AS is an independent network under the control of an operator, and the Internet consists of a number of interconnected ASes. If the same AS appears on the path between the client and its entry guard and also appears on the path between the exit and the destination, the AS can use traffic analysis to deanonymize the user.

This attack was originally examined by Feamster and Dingledine [2004] against Tor and Mixmaster [Möller et al. 2003]. Murdoch and Zeliniski further examined the threat with the respect to a more powerful attacker that has access to an Internet Exchange Point (IXP), which is an intersection point between ASes allowing the attacker to glean the network traffic information from multiple sources. They found that major IXP such as LINX (located in England) appeared on 27% of the paths. We next describe the most recent work carried out by Edman and Syverson [2009] with respect to this attack.

To understand the threats of an AS-level adversary, the authors used available routing information databases (RIBs) in order to construct AS-level graphs that depict ASes and their relationships and adjacencies using path inference algorithms. Then, they used a shortest path algorithm to compute paths between the ASes. Experimental results have shown that the probability that a single AS appears at the two ends of a circuit can be as high as 20%. This probability can be slightly decreased using an AS-aware or country-aware router path selection algorithm, such as LASTor. It is worth noting that the modifications in the router selection algorithm of Tor, such as enforcing a different /16 subnet for routers on a path, or the weighted bandwidth selection of routers, have had small improvements in limiting the threat of an AS-level adversary. For instance, the authors observed from their experiments that Tor's weighted router selection, with the different /16 subnets for routers enforced, had a end-to-end compromise rate of 17.8% with respect to the AS-level adversary, whereas uniform router selection yielded a 20.4% compromise rate.

10.1.2. Website Fingerprinting. Recall that Tor's anonymity protection fails against an adversary watching both the client and also the exit point of her circuit. A website fingerprinting adversary, on the other hand, only has to watch the client. As such, the client's ISP or national government is in an excellent position to mount this attack. To perform the attack, the adversary observes the packet counts, directions, and timings between the client and the anonymity system, and matches those against patterns indicative of particular websites of interest, using various machine learning techniques.

In the context of Tor, Herrmann *et al.*[2009] was the first to test the effectiveness of this attack against Tor. They used a multinomial Naive Bayes classifier, which was trained using features that are based on the frequency distribution of IP packet sizes and the flow direction. In a closed-world setting, where the classifier is trained to identify the website from a finite set of websites used for training, their classifier yielded only 3% success rate on Tor, despite the fact that they achieved more than 90% accuracy on `SSH` and `VPN` tunnels. The reason for the poor performance of this classifier on Tor is that it relies on packet sizes as a classification attribute, while cell sizes are fixed in Tor, resulting in nearly discrete distributions of packet sizes at the IP layer.

Panchenko *et al.*[2011] were able to increase the effectiveness of this attack using a Support Vector Machines (SVMs) classifier, using features that are mainly based on the volume, time, and direction of the traffic. In the closed-world setting (775 URLs), the authors use the same dataset used by Herrmann *et al.*and increase the detection rate to an alarming rate that exceeds 50%. Furthermore, the authors extend their experiments to an open-world setting where the user can access any URL he chooses. The attacker is assumed to be a regime that attempts to identify if a user is trying to access a censored website, so the classification in this setting is binary: allowed or prohibited website. Using different datasets for this setting, the authors are able to show high positive rates that range between 56% to 73%.

Several subsequent papers [Dyer et al. 2012; Cai et al. 2012; Wang and Goldberg 2013; Wang et al. 2014] demonstrated that it is possible to increase the accuracy using different classification techniques.Furthermore, To combat the website fingerprinting attack, several papers proposed and evaluated potential defenses [Dyer et al. 2012; Wang and Goldberg 2013; Cai et al. 2014a; Nithyanand et al. 2014; Wang et al. 2014; Cai et al. 2014b]. For example, Wang *et al.*[2014] reduce website download traces to packet sequences with direction and timing information. Their defence has two phases. First, the packet traces are clustered into a smaller number of anonymity sets, where every set contains very similar packet traces. Second, within every anonymity set, the authors find the shortest common supersequence (SCS), and pad all packet traces to that SCS. The goal is to make packet sequences within any anonymity set look exactly the same, while adding a minimum of overhead. In the live Tor network, one could implement such a defence at a guard node, protecting the traffic between the client and the guard node at a relatively low cost. However, to hedge against the guard node itself being the website fingerprinting adversary, having the exit of a circuit perform the defence, padding the traffic along the whole circuit from the exit all the way back to the client, is a somewhat higher-security, but higher-cost, alternative.

10.2. Path Selection Attacks

10.2.1. Selective Denial of Service (SDoS). The SDoS [Borisov et al. 2007] attack works by disrupting the reliability of the system with the goal of reducing its security. In this attack, the attacker denies service to circuits that he cannot compromise (by appearing at the two ends of a circuit). For example, if an entry guard is malicious, it will not allow its client to have a reliable anonymous communication except if the exit router is also a colluding node (an entry guard can determine if the exit node is a colluding node using traffic analysis). A circuit is reliable if all routers are reliable and either all routers are honest, or both the guard and the exit are compromised. Assuming all nodes are highly reliable in the system, the adversary can compromise as many as 50% of the circuits even when the fraction of dishonest nodes is as low as 20%.

10.2.2. Low-Resource Routing Attack. Bauer *et al.*[2007] present the low-resource routing attack, which works by exploiting the path selection algorithm of Tor and influencing it to select malicious routers. The adversary installs a number of low-resource nodes that falsely advertise high-bandwidth capabilities. When clients establish circuits, they will be trapped into selecting the malicious nodes with a higher probability because the router selection algorithm of Tor biases its selection towards higher-bandwidth routers.

To increase the effectiveness of the attack, malicious nodes perform a selective disruption where they refuse to relay traffic unless they are able to control the entry guard and the exit node of a circuit. The authors also describe an attack that enables the malicious routers to confirm that they are controlling the entry and exit positions of a circuit. To perform this attack, each malicious router logs some statistics and information regarding its connection, such as the IP addresses and ports and some connection timestamps, and reports the logs to a colluding centralized authority that runs the circuit linking analysis in real time. Experiments on an isolated Tor network have revealed the success of this attack. For example, if an attacker controls 10% of the network, it can compromise as many as 47% of the constructed paths.

In 2009, Tor deployed *bandwidth authorities* whose duty is to collectively measure the bandwidth of ORs. Those measurements are used to compute bandwidth weights that are published in the consensus document (described in Section 2.1). While such measurements provide a layer of protection against routing attacks, an adaptive malicious OR might still be able to game the measuring authorities in able to exaggerate its bandwidth.

10.2.3. Sniper Attack. The sniper attack [Jansen et al. 2014b] exploits a design weakness in Tor that allows a misbehaving client to successfully perform a DoS attack by exhausting an OR's memory, causing the Tor process either to be killed or to deny clients from forming circuits. In the basic form of this attack, the attacker uses his client to create a circuit through a malicious exit, placing the victim OR in the entry position. When the malicious client signals the exit to download a large file, the exit ignores Tor's window-based throttling mechanism, and keeps sending even if the window is empty. The malicious client does not read data from the entry, causing the cells to pile up in the circuit queue at the target router, eventually exhausting memory, and terminating the Tor process.

In the more efficient form of the attack, the adversary does not use a malicious exit, but sends circuit_sendme messages at a rate r that ensures that the circuit window does not exceed 1000 cells, a situation in which the exit terminates the circuit. To measure r, the adversary uses another client (or more), which builds another circuit through the same nodes and estimates the largest r that does not cause the exit to terminate the circuit. To speed up the attack, one or more clients can be used in parallel on the target. Jansen *et al.*show that this attack can disable the top 20 exit routers in only 29 minutes.

The adversary can also use this attack to speed up deanonymization of a hidden service (HS). The attacker deploys two nodes, one of which has to be a guard, and the other will serve as a rendezvous point (recall that this is an OR that the client chooses as a meeting point, causing the HS to build a circuit to it.) The attacker uses his client to repeatedly request connections to the HS. Using a previously proposed attack [Biryukov et al. 2013], the attacker can identify being one hop away from the HS by sending a crafted signal from the rendezvous OR towards the HS. That way, the attacker can identify all the guards of the HS. Next, the attacker uses the sniper attack to disable them, causing the HS to choose a new guard. Repeating this process increases the chances that the malicious guard is selected, and thereby breaking the anonymity of the HS.

10.3. Side Channel Information Attacks

10.3.1. Throughput Fingerprinting. One of the problems facing Tor is the heterogeneity of its volunteer-provided resources. Circuits built through different ORs have distinctive characteristics that can be enough to fingerprint the users building them. Mittal *et al.*[2011a] present a number of attacks based on fingerprinting circuit *throughput*. The insight of their attacks is that if two circuits share the same path or even just the bottleneck node on the path, observations of their throughput would be highly correlated. This allows an adversary to passively identify if two circuits share a subpath or a bottleneck. Also, an attacker can confirm if a specific router R carries a flow f by probing R (measuring R's throughput by creating a single-hop circuit through it and transferring data) and computing the throughput correlation between f and R. Furthermore, two separate malicious servers can confirm if two streams belong to the same circuit. If they indeed belong to the same circuit, the adversary learns that both streams belong to the same user. These attacks yield accurate results; however, they are costly. The cost of the attack scales as $\Theta(N)$ for the probing operations, which must be performed throughout the duration of the communication, where N is the number of possible routers.

10.3.2. Congestion Attacks. The first low-cost congestion attack on Tor was described by Murdoch and Danezis [2005]. The main contribution of this work is the realization that a global adversary is not necessary to perform traffic analysis attacks on Tor. The adversary in this attack can be a malicious server interested in learning the identities of its clients. When a client connects to the malicious server, the server responds to the client with data modulated in a very specific traffic pattern. The adversary then can learn the routers on the path by performing probing tests on the suspected routers (using a client). Experimental results on live Tor nodes showed that when a router in the client's circuit is probed, that router exhibited a probe latency that was highly correlated with the modulated traffic. The authors witnessed a very high success rate with few false positives. This attack was carried out in 2005, when Tor consisted only of 50 nodes, and was not as heavily used as today.

Another congestion attack was introduced by Evans *et al.*[2009], who observed that as the Tor network had grown over the years, the above attack of Murdoch and Danezis had ceased to be effective. The goal of this attack is simply to identify the entry guard of a client. The attack is carried out as follows: a client connects to a server using a malicious exit router, which injects JavaScript code into the client's requested web page. Next, since many Tor users do not disable JavaScript, the script would generate a signal with a specific pattern through the client's circuit, and thereby keep it alive. The attacker monitors the arrivals of the requests at the server, and records a baseline latency. The attacker then constructs a long circuit (preferably high bandwidth) that passes through the target suspected router many times. From the target router's point of view, this long path is multiple different circuits. Then, the long path circuit is used to congest the target router and if the target router is indeed the client's entry guard, the malicious server will observe a correlation between the latency of the signal and the duration of the congestion attack.

10.3.3. Network Latency. In the examples of side-channel information attacks described above, the goal of the adversary is to identify the Tor routers in a circuit, or to compromise the unlinkability of streams. Hopper *et al.*[2010] present two attacks that aim to reduce the anonymity of clients.

In the first attack, known as the *circuit linkability* attack, two malicious servers seek to find out if two different connections coming from one exit router belong to the same client or not. The distribution of the latency of each connection is measured (from the common exit router to the client). If the attacker observes the same distribution for the two connections, he assumes that the two connections come from the same circuit.

The second attack aims to approximate a client's location using a combination of the Murdoch-Danezis low-cost congestion attack and a latency attack. First, when a client is communicating through the Tor network, the congestion attack is carried out in order to identify the routers used in the circuit, and in particular to identify the entry guard of the client. The adversary's next goal is to measure the latency between the victim's client and its entry guard. This can be estimated by using a colluding Tor client to construct an identical circuit and measure the latency of the circuit to infer the latency of the link in question. Both attacks presented in this work have been tested and they both are successful in reducing the entropy of the anonymity set distribution. The authors suggest that to mitigate such attacks, it may be necessary to introduce artificial delays or use an enhanced path selection algorithm.

11. UNRESOLVED AND ONGOING ISSUES

In this section, we discuss a variety of open issues that call for further investigation in Tor. We begin by discussing the threat of Tor-based botnets, and then look at ongoing work on blocking resistance. We outline open problems and unresolved issues regarding hidden services, and argue for a more rigorous exploration for the timing analysis problem in Tor. Finally, we present future directions needed to improve Tor's performance.

11.1. Botnets

Botnets, large groups of malware-infected machines that are controlled by a single entity, known as the *botmaster*, have been growing in their stealth, sophistication and resiliency. Botnets started with a simple centralized architectural design where all bots (infected machines) connect to a single server, known as the *Command and Control* (C&C), to receive their instructions (such as to wage a DDoS attack), or to send the private data they harvest from their hosts. Because taking the botnet down is only as difficult as taking its C&C down, botnets evolved to more sophisticated P2P architectures, where the C&C remains more hidden. Researchers and security firms have devised and implemented techniques to detect P2P botnets and take them down [Nagaraja et al. 2010; Rossow et al. 2013].

The arms race still goes on. Various types of botnets have been recently found hiding behind Tor [Constantin 2012; Dunn 2013; Gottesman 2014]. Tor provides an attractive hideout for malware because a C&C can be deployed as a hidden service with a specific onion address that other bots are configured with, making the takedown operation difficult. Such botnets can cause a significant degradation on the performance of Tor [Hopper 2014]. For example, the 2013 Mevade/Sefnit botnet caused a spike (close to 600% increase) in the number of clients, which overloaded the network by requesting C&C descriptors and creating circuits. In fact, most hidden service queries and circuits might be related to botnets; Biryukov *et al.*[2014] analyzed the popular hidden services by looking at how often their descriptors are queried, and they found that the descriptors of the Skynet botnet were the most popular.

Detecting and removing botnets is a difficult problem in anonymity networks like Tor because (1) clients and hidden services are anonymized, and (2) traditional techniques [Antonakakis et al. 2010; Antonakakis et al. 2012; Bilge et al. 2014] that rely on detecting anomalies in DNS or IP traffic are not applicable to Tor since no DNS interactions occur within Tor to locate the hidden services, and because IP addresses are anonymized. Since only HSDir ORs are queried for the C&C hidden services' descriptors, one might consider monitoring the pattern and number of requests to HSDirs for different onion addresses as an approach to identify malicious HS addresses. Once suspicious onion addresses are detected, their responsible HSDirs can stop serving their descriptors; however, an attacker can embed his own HSDir OR to serve the descriptor of his malicious domain.[3] Furthermore, an attacker can periodically change the .onion address to avoid detection.

To protect Tor from abuses by botnets and malware, Hopper [2014] considers various approaches that are, on a high level, based on (1) increasing the cost of building circuits, and (2) throttling circuit building requests. He concludes that further evaluation is required before an approach is adopted, since those approaches can affect the experience of legitimate users, and may have anonymity implications if not studied carefully.

11.2. Blocking Resistance

Recall that The Tor Project introduced bridges as a means to resist the blocking of access to the Tor network. Those bridges are run by volunteers from around the world who can choose to keep their bridges private, where they only serve contacts in their social network, or to publish their contact information to the *bridge authorities*, which are servers responsible for distributing bridge information to Tor users worldwide.

When bridges were first introduced, bridge authorities would provide 3 bridges to each 24-bit IP prefix each day from its https server, so that a censor could not simply ask for the list of all bridges and block them. Ling *et al.*[2012] showed that they were able to harvest 2365 distinct bridge IPs during only 1 month. This was done by requesting bridges by sending emails and contacting the https server from PlanetLab nodes that have different /24 subnets; indeed, the Tor developers have documented 10 different ways for an attacker to enumerate bridges [Dingledine 2011]. As a response, users are requested to solve CAPTCHAs before receiving bridge IPs from the https server. While CAPTCHAs can slow down a bridge enumeration attack, alternative approaches should still be explored. This is a difficult problem: how can we distribute bridges at faster rate for legitimate users while enforcing a slower rate for censors?

In addition to blocking by IP address, censors started blocking by traffic flows. ISPs at censoring countries use DPI in order to identify Tor flows. For instance, the Great Firewall of China (GFC) was able to identify Tor traffic because the list of ciphers provided in its TLS Hello message is unique and fingerprintable [Winter and Lindskog 2012]. Various pluggable transports have been deployed to solve this problem by transforming Tor flows to mimic other protocols [Moghaddam et al. 2012; Fifield et al. 2012; Dyer et al. 2013]. However, even with pluggable transports, it is still possible to differentiate between real application traffic and a Tor-based imitation [Houmansadr et al. 2013]. There is currently no evidence that censors are actively seeking to identify pluggable transport traffic, and so users should be able to continue using them. That said, research will be needed in the future to devise different approaches or make current approaches more resistant to DPI.

11.3. Security of Hidden Services

Security improvements are needed for the design of hidden services in Tor. Biryukov *et al.*[2013] demonstrate that it is possible to enumerate the onion addresses of hidden services by deploying a hidden service directory, which is relatively easy to get, since (at the time of writing) a router must have an uptime of only 25 hours to gain the “HSDir” flag. This malicious directory will be responsible for any hidden service descriptor with an ID that is close to its fingerprint. Furthermore, because descriptor IDs have a validity period, the malicious directory will see new descriptors periodically. In fact,

[3]Recall that the HSDir responsible for serving a specific descriptor is computed deterministically.

the adversary can speed the attack by deploying multiple directories over different regions of the DHT ring, or by changing the router's fingerprint.

Not only is a directory able to enumerate onion addresses this way, but the attacker will be able to track the popularity of certain hidden services based on clients' request statistics. Worse, a malicious directory can try to target a specific hidden service in order to control its availability. To solve these problem, it is crucial to explore how to qualify a router to be a directory. One naive approach would be to harden the requirements needed to be a directory. Requiring more uptime increases the cost for the adversary as the deployed router needs to spend bandwidth relaying traffic during its uptime, but this can also slow honest routers from serving as directories. Exploring these tensions is needed for successful utilization of hidden services. The Tor Project is currently researching techniques to address those weaknesses in the HSDir system. [Dingledine 2013]

Finally, another privacy and anonymity concern for hidden services is anonymous payments. Many hidden services hinder their anonymity (and their clients' anonymity) by accepting donations and payments through the Bitcoin network. Since all Bitcoin transactions are publicly available in a public ledger, known as the *blockchain*, they may be traceable [Miers et al. 2013]. While there have been research proposals to make Bitcoin anonymous [Miers et al. 2013; Ben-Sasson et al. 2014], those are not deployed yet. There is a need to explore anonymous payment methods for hidden services.

11.4. Traffic and Timing Analysis

There are many questions surrounding the security assumptions and threat model in Tor that deserve more investigation. First, previous Tor research always assumed that if the attacker controlled the two ends of the circuit, it can perform traffic analysis to link the source to the destination. While this claim is plausible, the extent of the compromise rate is not understood and its costs have never been examined. With the increasing number of users and with the tremendous amount of traffic, it is possible that the end-to-end compromise rate may not be as high as previously assumed, or it could be difficult or costly for the attacker to execute. More experimental analysis for the end-to-end compromise rate assumption should be carried out.

If the Tor network is indeed vulnerable to end-to-end compromise as previously thought, then one important research direction is to devise countermeasures. While padding schemes have been proposed for general low-latency mix networks [Shmatikov and Wang 2006; Wang et al. 2008], such solutions are not effective in the context of Tor since an adversary can introduce timing watermarks in the traffic [Houmansadr and Borisov 2013], by crafting a specific delay pattern, or even by dropping packets. This led researchers to design countermeasures for timing attacks for low-latency anonymity networks, an example of which is the *layered mesh topology* [Feigenbaum et al. 2010], where the client constructs a circuit with several entry nodes, several middle nodes, and a single exit node. The client sends each cell to *all* entry routers. Each entry router then propagates the cells to all middle routers, and so on. Cells have timestamps specifying when a router has to send the cell to the next layer of routers. Finally, the last router on the path receives the cells and sends the data to the destination. If the adversary delays a cell, another honest router from the same layer (on a different path) will forward the replicated cell to the next layer of routers on time. This reduces the probability that the delay will be visible when the cells arrive at the last router. The problem of this approach is its high bandwidth and latency costs if adopted for Tor.

More approaches need to be explored and evaluated for Tor. A solution for this problem will provide a huge improvement for the anonymity of Tor and may help thwart serious threats such as the AS-level adversary describe above, since it is also based on end-to-end timing and traffic analysis.

11.5. Performance

From previous discussions, it can be seen that, despite previous research proposals (surveyed earlier), scalability problems are still lurking in the future of Tor. Proposed P2P proposals can not be adopted because (1) their lookup process reveals circuit information, and (2) they are susceptible to attacks where the adversary controls a large fraction of the network by introducing bogus nodes (using a botnet, for example). PIR approaches look promising, but they still need further investigation. PIR-Tor, for example, requires node reuse in its CPIR instantiation, lowering the security of Tor, while in its IT-PIR instantiation, requires multiple guards for each user to act as PIR servers. This creates tension with recent considerations to reduce the number of guards [Dingledine et al. 2014] to improve anonymity.

Providing incentives for users to run as routers can have a positive impact on scalability and congestion. As discussed in Section 5.1, incentive-based proposals suffer from shortcomings that need to be addressed. One promising direction is an approach based on *proof-of-bandwidth* like *torcoin* [Ghosh et al. 2014], where routers are rewarded with digital coins based on how much bandwidth they use relaying anonymous traffic. One challenge for a proof-of-bandwidth protocol is performing secure bandwidth measurements to ensure all network participants can easily verify that routers indeed spend what they claim to spend.

Furthermore, while there have been several transport layer proposals that aim to reduce congestion in Tor, it is still unclear what transport design provides the required trade-off between anonymity and performance for Tor. There is a need to experimentally compare the different transports under realistic user, network and traffic models that can emulate the real Tor network. Once a transport design is identified, a deployment plan must be carefully crafted in order to gradually and smoothly upgrade the network without denying service to its users.

12. CONCLUSION

In this paper, we identified key weaknesses in the design of Tor, the most widely used anonymous communication network, and accordingly classified previous proposals that aim to improve it. Previous work in this area can be categorized to proposals that aim to: (1) relieve network congestion (overlay and transport), (2) improve router selection, (3) enhance scalability, (4) reduce the communication/computational cost of circuit construction, and (5) improve its security. Within each of these categories, we surveyed the literature and compared the available techniques and shed light on their advantages and drawbacks in terms of anonymity, deployability and practicality. Finally, we discussed ongoing and unresolved issues that require further research and investigation.

Acknowledgement

This work was made possible by NPRP grant 7-1469-1-273 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors. The authors thank Prateek Mittal, Nikita Borisov, Rob Jansen, and other members of the Tor community for their useful feedback regarding this paper.

REFERENCES

Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. 2012. LASTor: A Low-Latency AS-Aware Tor Client. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*. IEEE Computer Society, Washington, DC, USA, 476–490.

Mashael AlSabah, Kevin Bauer, and Ian Goldberg. 2012. Enhancing Tor's Performance Using Real-Time Traffic Classification. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS '12)*. ACM, New York, NY, USA, 73–84.

Mashael AlSabah, Kevin S. Bauer, Tariq Elahi, and Ian Goldberg. 2013. The Path Less Travelled: Overcoming Tor's Bottlenecks with Traffic Splitting. In *Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings*. Springer, 143–163.

Mashael AlSabah, Kevin S. Bauer, Ian Goldberg, Dirk Grunwald, Damon McCoy, Stefan Savage, and Geoffrey M. Voelker. 2011. DefenestraTor: Throwing Out Windows in Tor. In *Privacy Enhancing Technologies - 11th International Symposium, PETS 2011, Waterloo, ON, Canada, July 27-29, 2011. Proceedings*. Springer Berlin Heidelberg, 134–154.

Mashael AlSabah and Ian Goldberg. 2013. PCTCP: Per-Circuit TCP-over-IPsec Transport for Anonymous Communication Overlay Networks. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. ACM, New York, NY, USA, 349–360.

Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. 2010. Building a Dynamic Reputation System for DNS. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*. USENIX Association, Berkeley, CA, USA, 273–290.

Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou II, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. 2012. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*. USENIX Association, Berkeley, CA, USA, 491–506.

Michael Backes, Aniket Kate, and Esfandiar Mohammadi. 2012. Ace: An Efficient Key-Exchange Protocol for Onion Routing. In *Proceedings of the 11th annual ACM Workshop on Privacy in the Electronic Society, WPES 2012, Raleigh, NC, USA, October 15, 2012*. ACM, New York, NY, USA, 55–64.

Kevin S. Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas C. Sicker. 2007. Low-Resource Routing Attacks Against Tor. In *Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society, WPES 2007, Alexandria, VA, USA, October 29, 2007*. ACM, New York, NY, USA, 11–20.

BBC News. 2007. Data disaster: Your Queries Answered. http://news.bbc.co.uk/2/hi/business/7105592.stm. (November 2007). Accessed March 2015.

Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, Washington, DC, USA, 459–474.

Leyla Bilge, Sevil Sen, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. 2014. Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains. *ACM Trans. Inf. Syst. Secur.* 16, 4 (2014), 14.

Alex Biryukov, Ivan Pustogarov, Fabrice Thill, and Ralf-Philipp Weinmann. 2014. Content and Popularity Analysis of Tor Hidden Services. In *34th International Conference on Distributed Computing Systems Workshops (ICDCS 2014 Workshops), Madrid, Spain, June 30 - July 3, 2014*. IEEE Computer Society, Washington, DC, USA, 188–193.

Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. 2013. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP '13)*. IEEE Computer Society, Washington, DC, USA, 80–94.

Henry Blodget. 2007. Compete CEO: ISPs Sell Clickstreams For $5 A Month. http://seekingalpha.com/article/29449-compete-ceo-isps-sell-clickstreams-for-5-a-month. (March 2007). Accessed March 2015.

Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. 2007. Denial of Service or Denial of Security?. In *Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*. ACM, New York, NY, USA, 92–102.

Torsten Braun, Christophe Diot, Anna Hoglander, and Vincent Roca. 1995. *An Experimental User Level Implementation of TCP*. Technical Report RR-2650. INRIA. http://hal.inria.fr/inria-00074040

Xiang Cai, Rishab Nithyanand, and Rob Johnson. 2014a. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society (WPES '14)*. ACM, New York, NY, USA, 121–130.

Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014b. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. ACM, New York, NY, USA, 227–238.

Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS '12)*. ACM, New York, NY, USA, 605–616.

Dario Catalano, Dario Fiore, and Rosario Gennaro. 2009. Certificateless Onion Routing. In *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. ACM, New York, NY, USA, 151–160.

David Chaum. 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM* 4, 2 (February 1981), 84–90.

Jen Christensen. 2008. FBI Tracked King's Every Move. http://edition.cnn.com/2008/US/03/31/mlk.fbi.conspiracy/. (December 2008). Accessed March 2015.

Lucian Constantin. 2012. Tor network used to command Skynet botnet. http://www.techworld.com/news/security/tor-network-used-command-skynet-botnet-3415592/. (December 2012). Accessed March 2015.

George Danezis, Claudia Díaz, and Carmela Troncoso. 2007. Two-Sided Statistical Disclosure Attack. In *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*. Springer, 30–44.

George Danezis, Roger Dingledine, and Nick Mathewson. 2003. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *2003 IEEE Symposium on Security and Privacy (S&P 2003), 11-14 May 2003, Berkeley, CA, USA*. IEEE Computer Society, Washington, DC, USA, 2–15.

Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. 2002. Towards Measuring Anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies (PET'02)*. Springer-Verlag, Berlin, Heidelberg, 54–68.

Tim Dierks and Eric Rescorla. 2008. RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2. http://www.ietf.org/rfc/rfc5246.txt. (August 2008). Accessed March 2015.

Roger Dingledine. 2011. Research Problems: Ten Ways to Discover Tor Bridges. https://blog.torproject.org/blog/research-problems-ten-ways-discover-tor-bridges. (October 2011). Accessed March 2015.

Roger Dingledine. 2013. Getting the HSDir flag should require more effort. https://trac.torproject.org/projects/tor/ticket/8243. (2013). Accessed March 2015.

Roger Dingledine, Nicholas Hopper, George Kadianakis, and Nick Mathewson. 2014. One Fast Guard for Life (or 9 Months). https://www.petsymposium.org/2014/papers/Dingledine.pdf. (July 2014). Accessed March 2015.

Roger Dingledine and Nick Mathewson. 2006. Anonymity Loves Company: Usability and the Network Effect. In *Workshop on the Economics of Information Security*. 547–559.

Roger Dingledine and Nick Mathewson. 2015a. Tor Directory Specification, version 3. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt. (2015). Accessed March 2015.

Roger Dingledine and Nick Mathewson. 2015b. Tor Protocol Specification. https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt. (2015). Accessed March 2015.

Roger Dingledine, Nick Mathewson, and Paul F. Syverson. 2004. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*. USENIX Association, Berkeley, CA, USA, 303–320.

John Dunn. 2013. Mevade botnet miscalculated effect on Tor network, says Damballa. http://www.techworld.com/news/security/mevade-botnet-miscalculated-effect-on-tor-network-says-damballa-3468988/. (September 2013). Accessed March 2015.

Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*. IEEE Computer Society, Washington, DC, USA, 332–346.

Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2013. Protocol Misidentification Made Easy with Format-Transforming Encryption. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. ACM, New York, NY, USA, 61–72.

Matthew Edman and Paul F. Syverson. 2009. As-Awareness in Tor Path Selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. ACM, New York, NY, USA, 380–389.

Aled Edwards and Steve Muir. 1995. Experiences Implementing a High Performance TCP in User-Space. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '95)*. ACM, New York, NY, USA, 196–205.

Nathan S. Evans, Roger Dingledine, and Christian Grothoff. 2009. A Practical Congestion Attack on Tor Using Long Paths. In *18th USENIX Security Symposium, Montreal, Canada, August 10-14, 2009, Proceedings*. USENIX Association, Berkeley, CA, USA, 33–50.

Stephen Farrell and Hannes Tschofenig. 2014. IETF BCP 188: Pervasive Monitoring Is an Attack. (May 2014). Accessed March 2015.

Nick Feamster and Roger Dingledine. 2004. Location Diversity in Anonymity Networks. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society (WPES '04)*. ACM, 66–76.

Joan Feigenbaum, Aaron Johnson, and Paul F. Syverson. 2010. Preventing Active Timing Attacks in Low-Latency Anonymous Communication. In *Privacy Enhancing Technologies, 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings*, Vol. 6205. Springer, 166–183.

David Fifield, Nate Hardison, Jonathan Ellithorpe, Emily Stark, Dan Boneh, Roger Dingledine, and Phillip A. Porras. 2012. Evading Censorship with Browser-Based Proxies. In *Privacy Enhancing Technologies - 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings*, Vol. 7384. Springer, 239–258. DOI:http://dx.doi.org/10.1007/978-3-642-31680-7_13

Michael J. Freedman and Robert Morris. 2002. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM, Washington, DC, 193–206.

John Geddes, Rob Jansen, and Nicholas Hopper. 2014. IMUX: Managing Tor Connections from Two to Infinity, and Beyond. In *Proceedings of the 13th annual ACM Workshop on Privacy in the Electronic Society (WPES '14)*. ACM, New York, NY, USA.

Mainak Ghosh, Miles Richardson, Bryan Ford, and Rob Jansen. 2014. A TorPath to TorCoin: Proof-of-Bandwidth Altcoins for Compensating Relays. http://www.robgjansen.com/publications/torpath-hotpets2014.pdf. (July 2014). Accessed March 2015.

Ian Goldberg, Douglas Stebila, and Berkant Ustaoglu. 2011. Anonymity and One-way Authentication in Key Exchange Protocols. University of Waterloo Technical Report CACR 2011-05. (May 2011).

Deepika Gopal and Nadia Heninger. 2012. Torchestra: Reducing Interactive Traffic Delays Over Tor. In *Proceedings of the 11th annual ACM Workshop on Privacy in the Electronic Society, WPES 2012, Raleigh, NC, USA, October 15, 2012*. ACM, New York, NY, USA, 31–42.

Yotam Gottesman. 2014. RSA Uncovers New POS Malware Operation Stealing Payment Card & Personal Information. https://blogs.rsa.com/rsa-uncovers-new-pos-malware-operation-stealing-payment-card-personal-information/. (January 2014). Accessed March 2015.

Larry Greenemeier. 2006. VA Secretary Comes Under Fire At House And Senate Data Theft Hearings. http://www.informationweek.com/va-secretary-comes-under-fire-at-house-a/188500312. (May 2006). Accessed March 2015.

Larry Greenemeier. 2008. Security Breach: Feds Lose Laptop Containing Sensitive Data — Again. http://www.scientificamerican.com/article.cfm?id=security-breach-lost-laptop. (March 2008). Accessed March 2015.

Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. 2009. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naive-bayes Classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security (CCSW '09)*. ACM, New York, NY, USA, 31–42.

Nicholas Hopper. 2014. Challenges in Protecting Tor Hidden Services from Botnet Abuse. In *Proceedings of Financial Cryptography and Data Security (FC'14)*. Springer, 316–325.

Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. 2010. How Much Anonymity Does Network Latency Leak? *ACM Trans. Inf. Syst. Secur.* 13, 2, Article 13 (March 2010), 28 pages.

Amir Houmansadr and Nikita Borisov. 2013. The Need for Flow Fingerprints to Link Correlated Network Flows. In *Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings*, Vol. 7981. Springer, 205–224.

Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. 2013. The Parrot Is Dead: Observing Unobservable Network Communications. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*. IEEE Computer Society, Washington, DC, USA, 65–79.

Rob Jansen, John Geddes, Chris Wacek, Micah Sherr, and Paul F. Syverson. 2014. Never Been KIST: Tor's Congestion Management Blossoms with Kernel-Informed Socket Transport. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.* USENIX Association, Berkeley, CA, USA, 127–142.

Rob Jansen and Nicholas Hopper. 2012. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society.

Rob Jansen, Nicholas Hopper, and Yongdae Kim. 2010. Recruiting new Tor Relays with BRAIDS. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*. ACM, New York, NY, USA, 319–328.

Rob Jansen, Nicholas Hopper, and Paul F. Syverson. 2012. Throttling Tor Bandwidth Parasites. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society.

Rob Jansen, Aaron Johnson, and Paul F. Syverson. 2013. LIRA: Lightweight Incentivized Routing for Anonymity. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*. The Internet Society.

Rob Jansen, Andrew Miller, Paul Syverson, and Bryan Ford. 2014a. From Onions to Shallots: Rewarding Tor Relays with TEARS. https://petsymposium.org/2014/papers/Jansen.pdf. (July 2014). Accessed April 2015.

Rob Jansen, Florian Tschorsch, Aaron Johnson, and Björn Scheuermann. 2014b. The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2013*. The Internet Society.

Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. 2007. Pairing-Based Onion Routing. In *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*. Springer, 95–112.

Jeremy Kirk. 2013. Passwords reset after Pony botnet stole 2 million credentials. http://www.pcworld.com/article/2069260/passwords-reset-after-pony-botnet-stole-2-million-credentials.html. (December 2013). Accessed March 2015.

Zhen Ling, Junzhou Luo, Wei Yu, Ming Yang, and Xinwen Fu. 2012. Extensive Analysis and Large-Scale Empirical Evaluation of Tor Bridge Discovery. In *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*. IEEE Computer Society, Washington, DC, USA, 2381–2389.

Mary Madden. 2014. Public Perceptions of Privacy and Security in the Post-Snowden Era. http://www.pewinternet.org/2014/11/12/public-privacy-perceptions/. (November 2014). Accessed March 2015.

Nayantara Mallesh and Matthew Wright. 2010. The Reverse Statistical Disclosure Attack. In *Information Hiding*, Rainer Böhme, Philip Fong, and Reihaneh Safavi-Naini (Eds.). Springer Berlin / Heidelberg, 221–234.

Damon McCoy, Kevin S. Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas C. Sicker. 2008. Shining Light in Dark Places: Understanding the Tor Network. In *Privacy Enhancing Technologies, 8th International Symposium, PETS 2008, Leuven, Belgium, July 23-25, 2008, Proceedings*. Springer, 63–76.

Jon McLachlan, Andrew Tran, Nicholas Hopper, and Yongdae Kim. 2009. Scalable Onion Routing with Torsk. In *Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. ACM, New York, NY, USA, 590–599.

Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. 2013. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*. IEEE Computer Society, Washington, DC, USA, 397–411.

Prateek Mittal and Nikita Borisov. 2009. ShadowWalker: Peer-to-peer Anonymous Communication Using Redundant Structured Topologies . In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*. ACM, New York, NY, USA, 161–172.

Prateek Mittal, Ahmed Khurshid, Joshua Juen, Matthew Caesar, and Nikita Borisov. 2011a. Stealthy Traffic Analysis of Low-Latency Anonymous Communication using Throughput Fingerprinting. In *Proceedings of the 18th ACM conference on Computer and Communications Security (CCS '11)*. ACM, New York, NY, USA, 215–226.

Prateek Mittal, Femi Olumofin, Carmela Troncoso, Nikita Borisov, and Ian Goldberg. 2011b. PIR-Tor: Scalable Anonymous Communication Using Private Information Retrieval. In *Proceedings of the 20th USENIX Conference on Security (SEC'11)*. USENIX Association, Berkeley, CA, USA, 31–31. http://dl.acm.org/citation.cfm?id=2028067.2028098

Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *Proceedings of the 19th ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*. ACM, New York, NY, USA, 97–108.

Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. 2003. Mixmaster Protocol - Version 3. IETF Internet Draft. (2003). http://www.eskimo.com/~rowdenw/crypt/Mix/draft-moeller-v3-01.txt

W. Brad Moore, Chris Wacek, and Micah Sherr. 2011. Exploring the Potential Benefits of Expanded Rate Limiting in Tor: Slow and Steady Wins the Race with Tortoise. In *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC)*. ACM, New York, NY, USA, 207–216.

Steven J. Murdoch and George Danezis. 2005. Low-Cost Traffic Analysis of Tor. In *2005 IEEE Symposium on Security and Privacy (S&P 2005), 8-11 May 2005, Oakland, CA, USA*. IEEE Computer Society, Washington, DC, USA, 183–195.

Steven J. Murdoch and Robert N. M. Watson. 2008. Metrics for Security and Performance in Low-Latency Anonymity Systems. In *Privacy Enhancing Technologies, 8th International Symposium, PETS 2008, Leuven, Belgium, July 23-25, 2008, Proceedings*. Springer, 115–132.

Steven J. Murdoch and Piotr Zielinski. 2007. Sampled Traffic Analysis by Internet-Exchange-Level Adversaries. In *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*. Springer, 167–183.

Kate Murphy. 2012. How to Muddy Your Tracks on the Internet. http://www.nytimes.com/2012/05/03/technology/personaltech/how-to-muddy-your-tracks-on-the-internet.html?_r=0. (May 2012). Accessed March 2015.

Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. 2010. BotGrep: Finding P2P Bots with Structured Graph Analysis. In *Proceedings of the 19th USENIX Conference on Security (USENIX Security'10)*. USENIX Association, Berkeley, CA, USA, 7–7.

Tsuen-Wan Ngan, Roger Dingledine, and Dan S. Wallach. 2010. Building Incentives into Tor. In *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers*. Springer, 238–256.

Rishab Nithyanand, Xiang Cai, and Rob Johnson. 2014. Glove: A Bespoke Website Fingerprinting Defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society (WPES '14)*. ACM, New York, NY, USA, 131–134.

Michael F. Nowlan, David Isaac Wolinsky, and Bryan Ford. 2013. Reducing Latency in Tor Circuits with Unordered Delivery. In *Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet*. USENIX Association, Berkeley, CA.

Lasse Øverlier and Paul F. Syverson. 2006. Locating Hidden Servers. In *2006 IEEE Symposium on Security and Privacy (S&P 2006), 21-24 May 2006, Berkeley, California, USA*. IEEE Computer Society, Washington, DC, USA, 100–114.

Lasse Øverlier and Paul F. Syverson. 2007. Improving Efficiency and Simplicity of Tor Circuit Establishment and Hidden Services. In *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*, Vol. 4776. Springer, 134–152.

Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, WPES 2011, Chicago, IL, USA, October 17, 2011*. ACM, New York, NY, USA, 103–114.

Andreas Pfitzmann and Marit Hansen. 2008. Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management — A Consolidated Proposal for Terminology. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.31.pdf. (February 2008). Accessed March 2015.

Jean-François Raymond. 2000. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*. Springer-Verlag, LNCS 2009, 10–29.

Joel Reardon and Ian Goldberg. 2009. Improving Tor using a TCP-over-DTLS Tunnel. In *18th USENIX Security Symposium, Montreal, Canada, August 10-14, 2009, Proceedings*. USENIX Association, Berkeley, CA, USA, 119–134.

Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. 1998. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications* 16, 4 (1998), 482–494.

Marc Rennhard and Bernhard Plattner. 2002. Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection. In *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society (WPES '02)*. ACM, New York, NY, USA, 91–102.

Christian Rossow, Dennis Andriesse, Tillmann Werner, Brett Stone-Gross, Daniel Plohmann, Christian J. Dietrich, and Herbert Bos. 2013. SoK: P2PWNED - Modeling and Evaluating the Resilience of Peer-to-Peer Botnets. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP '13)*. IEEE Computer Society, Washington, DC, USA, 97–111.

Douglas Rushkoff. 2012. Will Your Internet Provider Be Spying on You? http://www.cnn.com/2012/07/06/opinion/rushkoff-online-monitoring/. (July 2012). Accessed March 2015.

Juha Saarinen. 2014. First Shellshock Botnet Attacks Akamai, US DoD Networks. http://www.itnews.com.au/News/396197,first-shellshock-botnet-attacks-akamai-us-dod-networks.aspx. (September 2014). Accessed March 2015.

Max Schuchard, Alexander W. Dean, Victor Heorhiadi, Nicholas Hopper, and Yongdae Kim. 2010. Balancing the Shadows. In *Proceedings of the 9th annual ACM Workshop on Privacy in the Electronic Society (WPES '10)*. ACM, New York, NY, USA, 1–10.

Symantec Security. 2014. Apple IDs Targeted by Kelihos Botnet Phishing Campaign. http://www.symantec.com/connect/blogs/apple-ids-targeted-kelihos-botnet-phishing-campaign. (September 2014). Accessed March 2015.

Andrei Serjantov and George Danezis. 2002. Towards an Information Theoretic Metric for Anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies (PET'02)*. Springer-Verlag, Berlin, Heidelberg, 41–53.

Micah Sherr, Matt Blaze, and Boon Thau Loo. 2009. Scalable Link-Based Relay Selection for Anonymous Routing. In *Privacy Enhancing Technologies, 9th International Symposium, PETS 2009, Seattle, WA, USA, August 5-7, 2009. Proceedings*. Springer, 73–93.

Vitaly Shmatikov and Ming-Hsiu Wang. 2006. Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In *Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006, Proceedings*, Vol. 4189. Springer, 18–33.

Robin Snader. 2010. *Path Selection for Performance- and Security-Improved Onion Routing*. Ph.D. Dissertation. University of Illinois at Urbana-Champaign.

Robin Snader and Nikita Borisov. 2008. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*. The Internet Society.

Can Tang and Ian Goldberg. 2010. An Improved Algorithm for Tor Circuit Scheduling. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*. ACM, New York, NY, USA, 329–339.

The Tor Project. 2015a. Tor Metrics Portal: Network. https://metrics.torproject.org/networksize.html?graph=networksize&start=2014-12-08&end=2015-03-08. (October 2015). Accessed March 2015.

The Tor Project. 2015b. Tor Metrics Portal: Network. https://metrics.torproject.org/torperf.html. (October 2015). Accessed March 2015.

Florian Tschorsch and Björn Scheurmann. 2011. Tor is Unfair — And What to Do About It. In *Proceedings of the 36th IEEE Conference on Local Computer Networks (LCN)*. IEEE Computer Society, Washington, DC, USA, 432–440.

Camilo Viecco. 2008. UDP-OR: A Fair Onion Transport Design. http://www.petsymposium.org/2008/hotpets/udp-tor.pdf. (July 2008). Accessed March 2015.

Chris Wacek, Henry Tan, Kevin S. Bauer, and Micah Sherr. 2013. An Empirical Evaluation of Relay Selection in Tor. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*. The Internet Society.

Qiyan Wang, Prateek Mittal, and Nikita Borisov. 2010. In Search of An Anonymous and Aecure Lookup: Attacks on Structured Peer-to-Peer Anonymous Communication Systems. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*. ACM, New York, NY, USA, 308–318.

Tao Wang, Kevin S. Bauer, Clara Forero, and Ian Goldberg. 2012. Congestion-Aware Path Selection for Tor. In *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, Vol. 7397. Springer, 98–113.

Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.* USENIX Association, Berkeley, CA, USA, 143–157.

Tao Wang and Ian Goldberg. 2013. Improved Website Fingerprinting on Tor. In *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*. ACM, 201–212.

Wei Wang, Mehul Motani, and Vikram Srinivasan. 2008. Dependent link padding algorithms for low latency anonymity systems. In *Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*. ACM, 323–332.

Philipp Winter and Stefan Lindskog. 2012. How the Great Firewall of China is Blocking Tor. In *2nd USENIX Workshop on Free and Open Communications on the Internet, FOCI '12, Bellevue, WA, USA, August 6, 2012*. USENIX Association, Berkeley, CA, USA. https://www.usenix.org/conference/foci12/workshop-program/presentation/winter

Matthew K. Wright, Micah Adler, Brian Neil Levine, and Clay Shields. 2004. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Trans. Inf. Syst. Secur.* 7, 4 (2004), 489–522.