A Survey on Tor and I2P

Bernd Conrad and Fatemeh Shirazi

Department of Computer Science, TU Darmstadt
Darmstadt, Germany
Email: {bconrad,fshirazi}@cdc.informatik.tu-darmstadt.de

***Abstract*—This paper gives a short introduction and a comparison on two low-latency anonymous communication networks. The main part features a review of the low latency anonymous communication networks, namely, The Onion Routing (Tor) and the Invisible Internet Project (I2P). An introduction to their overall structure is given, followed by a detailed description of the core parts of both networks. Furthermore, a comparison of both will feature important aspects like node selection, performance and scalability. The detailed description and comparison of the two systems show that determining which system to use highly depends on the field of application, since each system has its strength and weaknesses in specific areas.**

***Keywords*–Tor; I2P; low latency anonymous communication networks.**

I. Introduction

When communicating over the Internet, IP addresses are used to provide a unique identifier to address each party. Even if a message is encrypted to protect the data content, source and destination address are contained in clear in the corresponding IP datagram headers, otherwise messages could not be routed to their destination. Thus, communication over the Internet is not anonymous. An adversary monitoring the network traffic could easily identify two parties communicating with each other. Anonymous Communication Networks (ACNs) are an essential building block for protecting privacy online, as they enable users to communicate anonymously over the Internet. Using the ACN, users can conceal the destination of their communications towards local adversaries, e.g., their ISP, as well as protect their identity towards the destination itself, e.g., a website. Typically, an ACN is an overlay network composed of a set of routers (also-called relays, or *nodes*), in which packets are relayed using multiple routers to achieve anonymity. In general, anonymous communication networks can be divided into two main categories; *high latency anonymous communication networks* in which it takes a relatively longer time for the message to travel through the network and reach it's destination, usually ranging from a few hours to several days [1]. This delay is tolerable when using those systems for non-interactive applications like email, however today most applications on the Internet are real-time, interactive applications that require a low latency, e.g., web browsing. Systems designed to provide anonymity and low latencies when using real-time, interactive applications are called *low-latency anonymous communication networks*. In this paper, we review and compare Tor [2][3] and *Invisible Internet Project* (I2P) [4], which are currently among the most commonly used low latency anonymous communication networks. Other examples of practical anonymous communication networks are Freenet [5][6], JAP [7][8], and GNUnet [9][8]. Regardless of several similarities, Tor and I2P have noticeable differences, which makes them preferable for specific usages. In order to be able to decide on which one of them to use, one has to fully understand these differences. In this paper, we review some of the main differences between Tor and I2P.

The remaining paper is organized as follows: A short introduction to the anonymous communication network Tor is given in Section 2. Section 3 describes I2P and its core elements. Finally, a comparison of both systems will be presented in Section 4. Section 5 concludes.

II. Onion routing and Tor

One approach to achieve low latencies and at the same time protect against a strong adversary is the arguably most prevalent onion routing design [1], a distributed overlay network designed to anonymize TCP-based applications [3]. According to Danzis and Diaz [10] “the objective of onion routing is to make traffic analysis harder for an adversary, as well aims first at protecting the unlinkability of two participants who know each other from third parties, and secondly, at protecting the identities of the two communicating parties from each other”.

A set of servers called Onion Routers (OR) are used to relay messages. Each OR maintains a private and public key pair, while the public part should be known to all clients wishing to participate in the network. Clients choose an ordered sequence of ORs they want to use to relay their data and establish a so-called *circuit*, a bidirectional tunnel. This method is called onion encryption and will be described more precisely later on. Each layer contains a symmetric key, a label and addressing information about the next OR. Messages sent through circuits are also onion encrypted, this time using the symmetric key of each OR [10]. Each OR is only able to remove the corresponding layer of encryption and forwards the message to the next OR in the circuit. The last OR in the circuit is able to forward the message to its destination. The potential response of the receiver is sent to the last OR in the circuit and is relayed back to sender through the exact same circuit. This time, each OR adds a layer of encryption to the message. Hence, another onion encrypted message is constructed that only the sender is able to decrypt and therefore recover the response. An important fact in regards to anonymity and security is that only the first OR in a circuit knows the IP address of the client, and only the last OR of a circuit knows the receiver of a message. All intermediate ORs only know its predecessor and its successor, without even knowing which other ORs are participating in the circuit [3]. A circuit may be used to relay multiple messages from a single application [1], but each TCP stream needs its own circuit [3]. The ORs implement a very close to first-in first-out mixing strategy to provide low latency. This makes onion routing susceptible to a number of attacks. Due to missing cover traffic, an adversary may use traffic analysis and timing attacks to monitor a traffic pattern, follow the message stream and identify communicating parties [10][11]. Nonetheless, onion routing is a promising design to provide a low latency anonymous communication network and many currently used systems are built upon this design.

A. Tor

Tor is a distributed-trust, circuit-based low latency anonymous communication network. It builds upon the onion routing design, but makes many modifications and improvements in regards to security, efficiency, and deployability [1]. The Tor network is an overlay network that uses a set of volunteer servers, called *Onion Routers* (OR), to build *circuits* and relay messages [11]. Each user runs a software called *Onion Proxy* (OP) that manages all Tor related processes, e.g., establishing circuits or handling connections from user applications [3]. To build a circuit, the OP select an ordered set of usually three ORs out of the set of all known ORs. The first OR in the set is called entry guard, the last is called exit router and all others are called intermediate routers [12][13]. The process of selecting ORs for a circuit is called *node selection* and will be described more precisely later. To obtain a list of all known ORs, a set of *directory authority servers* are used. After selecting a set of ORs, the OP contacts the entry guard and builds a circuit with it. This newly created circuit is used to contact the next OR to extend the circuit. This procedure is iteratively repeated until all ORs of the set are part of the circuit. The established circuit can now be used to anonymously relay messages. Messages are *onion encrypted* and only the exit router is able to access and forward a message to its destination.

Onion Router: Onion Routers (OR) are the core part of the network since they are necessary to build circuits. All ORs are connected with each other using Transport Layer Security (TLS) connections. This prevents an attacker from modifying data or impersonating an OR [3]. Each OR also maintains two keys: a long-term identity key, used to sign TLS certificates, *router descriptors*, and directories; and a short-term onion key, used to decrypt user requests to build a circuit and negotiate short-lived symmetric keys [3][11]. Router descriptors uniquely identify each OR and contain all relevant data to contact and list an OR: public keys, IP address, bandwidth, *exit policies*, and more [3]. Exit policies describe which hosts and ports the OR is willing to connect to, this is particularly important for the later described node selection process.

Directory Server: To be able to retrieve a list of all available ORs, authoritative directory servers distributing signed directories are used [12]. These servers need to be well-known, which means that the IP addresses of this servers are commonly known and/or published on specific websites, and able to track changes in network topology. The directory contains the router descriptor of each listed OR and a network status document. The network status document contains measured bandwidths of ORs. Only ORs that are verified via their identity key are listed in the directory, otherwise they are ignored. There are multiple directory servers to protect against active attacks against directory servers [3], e.g., potentially by Denial-of-Service (DoS) attacks, which prevents having a single point of failure. All directory servers also merge their known topology of the network with each other and release a common signed directory of the whole network. Directories are automatically fetched by the OP. The client software also contains a default list of directory servers [3].

Node selection: To guarantee a good performance and to prevent choosing a corrupted OR as entry guard, the Tor client uses a path selection algorithm to select the ORs used to build circuits [14]. All known ORs are categorized into three tiers:

- *Entry guard router*: Stable, fast and well-known ORs.
- *Intermediate router*: All known ORs.
- *Exit router*: ORs with matching exit policies.

The network status document and all router descriptors maintained by the directory servers are fetched by the OP. Both contain router bandwidth information. The router descriptor contains a self-advertised bandwidth and the network status document contains a value measured by the directory servers. As long as the measured value is available, it will be preferred due to the fact that self-advertised information are considered not trustworthy. The bandwidth information is used to select the intermediate and exit routers in a weighted probabilistic manner [14]. This means a router with a higher bandwidth is more likely to be chosen.

The OP maintains a list of three potential entry guards, chosen from a list of all ORs with a long uptime and known to be fast and stable [14]. The entry guard is then randomly chosen from this list of three entry guards, and used for all circuits. After normally 30 days, the list of three entry guards is rebuilt and a new entry guard is chosen. ORs serving as exit routers can also be considered as entry guards and intermediate routers, but “only if the available total bandwidth of exit nodes is at least one third of the overall available bandwidth of all routers”, also, to provide load balancing, “their probability of being chosen is lowered in a weighted way” [14].

Cell: Tor uses a special format, called cells, for all messages that are sent through the network. Cells have a fixed size of 512 bytes and consist of a header and a payload. Tor uses two kinds of cells, *control cells* and *relay cells*. Control cells are used to set up, maintain and destroy circuits. Relay cells are used to relay messages along the circuit. Relay cells also contain an additional header in front of the payload used to distinguish between different streams and to perform end-to-end integrity checking [3]. The additional header also allows the network to detect congestion or flooding, and therefore reduce outgoing traffic until the congestion subsides [3].

Circuit: A circuit is a bidirectional virtual connection set up between the OP and an ordered set of ORs. In contrast to onion routing, a single circuit can be used by multiple TCP streams at the same time. To prevent an adversary from linking streams together, the default circuit lifetime is 10 minutes. After this time, a circuit is destroyed and a newly built circuit is used. Building new circuits is done beforehand in the background, therefore no additional latency is generated [3].

Onion Encryption: After establishing a circuit, the OP can start sending data messages within relay cells. Similar to the onion routing design the header and payload of a cell is iteratively encrypted using the symmetric key of each OR participating in the circuit [3]. Starting with the key of the exit router, traversing the circuit using the key of each intermediate

OR until the entry guard is reached. The following example shows the encryption process for three ORs, whereas k_1 is the key of the entry guard, k_2 the key of the intermediate router and k_3 the key of the exit router. $E_{k_1}(E_{k_2}(E_{k_3}(cell)))$ This procedure is called *onion encryption*. As the cell moves along the circuit, one layer per OR is removed. Only the exit router is able to extract the destination address and the payload, which may be the actual plaintext or an end-to-end-encrypted message, and forwards the payload to its destination. The reply can only be sent back along the same circuit. Each OR in the circuit adds his layer of encryption, using his negotiated symmetric key, before relaying the cell to its predecessor. Only the OP is able to fully decrypt the onion encrypted reply, since he is the only one that knows all negotiated symmetric keys.

III. I2P

I2P is a message-oriented, peer-to-peer-based low latency anonymous communication network. The network was mainly designed to enable fully anonymous communication between two parties inside the network [15][16]. I2P was first proposed in 2003, having its roots in the Invisible Internet Project (IIP) [17][18]. A wide range of applications inside the I2P network are available, e.g., anonymous web-hosting, web browsing, file-sharing, email and many more. Using external services, meaning services that are not hosted within the I2P network, requires the use of an out-proxy [16]. At the time of writing, the I2P network consists of 23738 routers with an average count of 25687 routers [19].

I2P is an overlay network allowing users to anonymously interact within the network. Technically, I2P is a multi-application Java framework designed to provide anonymous P2P networking [20]. Each user is running a so-called I2P router, the core part of the I2P software. All messages are relayed through *tunnels* built by each I2P router using other I2P peers. Tunnels can only be used in one direction; therefore, tunnels for outgoing and incoming traffic need to be built, so-called *inbound* and *outbound* tunnels. The selection of peers is done via a *tier-based peer selection* algorithm running on each I2P router. After establishing inbound and outbound tunnels clients may publish their contact information in a global database, called *netDB*. The netDB contains contact information for each I2P peer and each publicly running service inside the I2P network. Messages sent through the I2P network are end-to-end encrypted using *garlic encryption*. Garlic encryption is very similar to onion encryption, with the difference that multiple data messages may be contained in a single *garlic message*. Therefore, a single garlic message may contain multiple messages for different recipients.

I2P Router: The I2P network is formed by peers (also-called clients, nodes or router) running the I2P software, allowing applications to communicate through the I2P network [16]. The core part of this software is the I2P router. The I2P router is responsible for maintaining peer statistics, which are required for the peer selection described later, performing cryptographic operations, building tunnels, providing services and relaying messages. Applications heavily rely on the tunnels built by the I2P router to remain anonymous [20].

NetDB, RouterInfo and LeaseSet: Super-peers, called *floodfill peers*, are used to build and manage a network database, called *netDB*. The netDB is based on a distributed hash table and contains all known information about the I2P network, therefore all I2P peers and services. Each floodfill peer is only responsible for information of a specific part of the network. The Kademlia XOR distance metric [21] is used to determine which part of the network a floodfill peer is responsible for, based on the peers ID [20]. Peers with sufficient bandwidth may get promoted to floodfill peers if the amount of available floodfill peers drops below a certain threshold [22]. The netDB stores two types of data, a *routerInfo* structure that describes an I2P peer and a *leaseSet* for each known service [23]. All I2P peers are identified by a data structure called *routerInfo*, containing all important information about the peer (IP address, port, peer ID, I2P stable version number, network version, transport capabilities and some statistical data [23]), his public key and a 256 bit hash-identifier. To retrieve an initial list of available I2P peers, a list of routerInfos can be downloaded from a non-anonymous, well-known web server. Retrieving the initial list of routerInfos is called *reseeding* [20][23].

A *leaseSet* is used to store information about how to contact an internal I2P service, called *destination*. The leaseSet specifies a set of entry points, called leases. A lease identifies a peer that serves as an inbound gateway to an inbound tunnel of the corresponding service [20]. Both, routerInfos and leaseSets, can easily be stored and retrieved by contacting the nearest floodfill peer. In case of storing, the floodfill peer will distribute the received routerInfo or leaseSet to the seven nearest floodfill peers. In case of retrieving, the two closest floodfill peers are contacted. If the requested information is not available, the floodfill peer replies with a list of other near floodfill peers. The peer keeps contacting floodfill peers until the needed information is retrieved or all floodfill peers have been contacted [20].

Destination: All destinations in the I2P network are identified by a 516 byte crypto key that consists of a 256-byte public key, a 128-byte signing key and a (currently unused) null certificate. A destination in I2P refers to an internal service provided by an I2P router. To map destination names to their crypto key, three local host files are used, similar to traditional DNS. To merge external and local host files, I2P provides an address book application [18][22]. This way of addressing each individual destination further increases the anonymity since it also decouples the service from the I2P router its provided by [16].

Tunnel: All messages in the I2P network are transmitted through so-called *tunnels*. A tunnel is a unidirectional encrypted virtual connection using typically 2 to 3 I2P peers [23][18]. Unlike Tor the I2P router seeking to establish a tunnel is also part of the tunnel. At startup each I2P router builds up some tunnels for incoming traffic, called inbound tunnel, and outgoing traffic, called outbound tunnel. The first I2P peer of a tunnel is called tunnel gateway, the last I2P peer of tunnel is called tunnel endpoint. For outbound tunnels, the I2P router that established the tunnel is always the gateway. For inbound tunnels, the I2P router that established the tunnel is always the endpoint. The default amount and length of tunnels can be specified by the user in the I2P settings. The length of a tunnel is a trade-off between performance and anonymity [20]. Longer tunnels increase the anonymity, while they decrease the performance and the other way round. An application is not bound to a specific tunnel and may use different tunnels to relay messages. There are two kinds of tunnels, *exploratory* and *client* tunnels [17]. Exploratory tunnels are low bandwidth tunnels and not used for privacy-sensitive operations. A router uses this tunnel to contact floodfill peers and retrieve the netDB. Exploratory tunnels are also used to build, manage and destroy other tunnels [23]. Client tunnels are used to relay application messages and retrieve leaseSets; therefore, are high bandwidth tunnels. Tunnels have a maximum lifetime of 10 minutes. After this period of time the tunnel is destroyed and a new one is used. Constantly rebuilding tunnels seeks to prevent traffic analysis attacks [16].

Tunnel Establishment: Building a new tunnel is done by first selecting an ordered set of I2P peers. This selection of peers is done with *tier-based peer selection* and *peer profiling* to categorize peers into tiers. An exploratory tunnel is used to send a single, multiple times encrypted *tunnel construction request* to the first I2P router. Every layer contains necessary information for each single I2P peer, e.g., symmetric key and successor address. Like in the original onion routing design the message is forwarded until it reaches the last I2P peer of the tunnel. The response is then routed back to the originator while each I2P peer adds a layer of encryption [24]. The receiving I2P peers are free to decide if they want to decline the request or accept to participate in the tunnel. An already established tunnel can still fail at any time if, e.g., the I2P peer is not able to handle the traffic or leaves the network (goes offline) [20].

Tier-based Peer Selection and Peer Profiling: Tier-based peer selection is the process of selecting peers used to build a tunnel based on tiers they are assigned to. *Peer profiling* is used to categorize peers into those tiers. Peers sharing a tier share certain performance characteristics [23]. Peer profiling is done by the I2P router, he keeps track of various performance statistics of other peers and maintains a database containing this statistics, called *profiles*. However, no active bandwidth probing or other actions that may generate non-data traffic are used. Every 30 seconds all profiles are sorted into three tiers based on various metric like speed and capacity [23][20]:

- *Not-failing*: All known peers. Typically 300-500 peers.
- *Well-integrated*: Peers that claim to know many other peers.
- *High-capacity*: Peers that are known to most likely accept tunnel build request. Typically 10-30 peers.
- *Fast*: Peers from the high-capacity tier with a high bandwidth. Typically 8-15 peers.

Note that all fast tier peers are always also high-capacity tier peers [23]. When constructing a client tunnel, peers from the fast tier are used. If no sufficient amount of fast tier peers is available, high-capacity tier peers are selected. High-capacity tier peers are used when constructing an exploratory tunnel. Both, the well-integrated and not-failing tier peers are fallback options, if no high-capacity and fast tier peers are available. However this is unlikely to happen in practice [20]. The actual selection of peers for exploratory tunnels is done using a weighted random function [23]. Also peers sharing the same /16 subnet will not be used together within the same tunnel [23].

Garlic Routing, Garlic Message and Garlic Encryption: When at least one outbound and one inbound tunnel is constructed, the I2P router is able to send and receive messages through the I2P network. To communicate with an I2P service, the router first needs to retrieve the destination of this service from a floodfill peer [17]. The destination specifies a set of inbound tunnel gateways of the corresponding service. I2P uses so-called *Garlic routing*, a variation of the onion routing design described in Section II. Garlic routing uses garlic messages that can contain multiple so-called cloves. Cloves are data messages with additional routing instructions like delays. This means a garlic message may contain multiple application messages. The actual data messages are end-to-end encrypted with the receiver's public key. The garlic message itself is encrypted multiple times using the symmetric keys negotiated with the tunnel peers [22][20]. When traversing the tunnel, each I2P peer removes one layer of encryption until the garlic message reaches the outbound tunnel endpoint. The outbound endpoint forwards each message to its destination's inbound tunnel gateway. The inbound gateway will forward the garlic message to the actual recipient while each peer participating in the tunnel adds a layer of encryption (using the negotiated symmetric keys). Only the recipient is able to remove all encryption layers of the garlic message as well as the end-to-end encryption of the data-messages [20][17]. As mentioned before, if a service outside the I2P network is addressed, an outproxy has to be used [18], although according to Schimmer et al. [23] "only one HTTP outproxy is publicly advertised and accessible". When using an outproxy, end-to-end encryption is, similar to Tor, not guaranteed, since it depends on the transport layer protocol that is used.

IV. Tor vs I2P

There are a few obvious difference between both networks. While Tor is relying on servers provided by volunteers to build circuits, I2P uses peers with sufficient performance characteristics participating in the network to build tunnels. Also, Tor is optimized and designed for exit traffic with a large number of exit routers, whereas I2P is designed to provide services inside the network and only features a small set of outproxies [25]. Nonetheless, both seek to provide strong anonymity with low latency when using real-time, interactive services. A comparison of a few important aspects of anonymous communication networks is presented as follows.

SOCKS vs I2P API: While this seems like a rather technical aspect, it greatly changes the effort and ability to build applications that use either the I2P or Tor network to anonymously communicate over the Internet. Tor uses the *Socket Secure* (SOCKS) interface and therefore SOCKS-aware applications may be easily pointed at the Tor software, which then handles everything else. Tor, in this case, acts as a proxy server. This means, applications able to use SOCKS can be used without any changes [3]. I2P, on the other hand, is a middleware providing APIs that applications can use to communicate through the network, meaning applications either need to be costly adjusted, if at all possible, or implemented from scratch. The use of SOCKS by Tor has two downsides:

1) The SOCKS interface is only able to transmit messages over TCP while I2P has the choice between UDP and TCP [25]. This may enable I2P to deliver better performance when using certain applications.
2) Messages sent by applications may still contain information that could identify the sender. To prevent this, application-level proxies with filtering features, e.g., Privoxy, need to be used [3].

ISBN: 978-1-61208-362-9

Available Applications: Both, I2P and Tor feature a wide range of applications, whereas most I2P applications are exclusively made to access services inside the I2P network, with some exceptions, e.g., Susimail/2IpMail is able to send and receive mails from the public Internet [18]. Tor on the other hand, due to the fact it is using the SOCKS interface as mentioned before, is able to be used with any application able to be configured using a SOCKS proxy, e.g., nearly every commonly used web browser.

Message Security and Anonymity: Both networks feature various layers of encryption, starting with transport layer encryption provided by the TLS connection maintained by the ORs or respectively I2P peers. I2P also features an additional tunnel encryption. Messages sent through the networks are either onion or Garlic encrypted. This means the connection from the user to the tunnel or circuit is always encrypted. As long as interacting inside the network, messages in I2P are also end-to-end encrypted. In the case of Tor, end-to-end encryption can not be guaranteed since it depends on the transport layer protocol that is used. Therefore, insecure protocols should not be used, as a corrupted exit node may record messages sent in plaintext and recover usernames and/or passwords [12]. In Tor only the first OR of a circuit knows the IP address of the actual user, all subsequent ORs only know its predecessor and successor. Also only the last OR in the circuit knows the actual receiver. Nonetheless, this is a potential risk since corrupted ORs may be able to link communicating parties together. Therefore, the user's anonymity highly depends on Tors node selection algorithm selecting trustworthy entry guards. In the case of I2P even the first peer does not know if it is forwarding the message for another peer that is part of the tunnel or the actual sender. Therefore, entry guards like in Tor are not necessary.

Performance: In 2011, Ehlert analyzed and compared the latency and bandwidth when accessing the public Internet with either I2P or Tor [17]. The latency when issuing simple HTTP-GET-Requests and the average latency when accessing whole web pages were recorded and evaluated, as well as the download speed when receiving files from a fixed location. While I2P was able to achieve better results when issuing simple HTTP-GET-Requests, Tor provides clearly better results in terms of accessing whole web pages and downloading files. In 50% of all cases Tor was able to retrieve a whole web page in less than 16.99 seconds, while 50% of the I2P request took up to 103.19 seconds. In case of download speed, Tor was able to deliver an average speed of 51.62 kB/s compared to the 12.91 kB/s of I2P. The author also seeks to explain why I2P is scoring better results than Tor when issuing simple HTTP-GET-Requests. He states, that the discrepancy may be explained with the good distribution of I2P peers in Europe and therefore good latencies when issuing simple HTTP-GET-Request. For further information see [17].

Scalability: Increasing the number of clients participating in the anonymous network directly influences both Tor and I2P. Although the anonymity set becomes larger and therefore stronger anonymity may be present, the network traffic increases and may cause problems like congestion. In case of Tor this means, the amount of routers used to build circuits may very likely need to be increased. This problem may get even worse due to the fact that only a small subset of all ORs is used as entry guards and exit routers. This may, depending on the amount of new clients joining the network, sooner or later lead to congestion problems and therefore increase the latency and decrease the available bandwidth. Congestion and high latencies will directly affect the user experience and network usability. Increasing the amount of ORs also serves another problem, the growing directory. On the one hand additional bandwidth is used to receive directories and on the other hand the effort to keep track of the whole network increases. As mentioned before, Tor also uses active bandwidth probing which additionally increases the traffic depending on the amount of new ORs joining the network. Also the assumption that every OR in the network is able maintain a direct connection to each other OR seems rather unlikely as the number of ORs increases [1][3]. In case of I2P, new peers joining the network may also be peers that can be used to build tunnels, assuming they provide enough capacity and bandwidth. Therefore, congestion is not likely to appear, however, if a sufficient number of clients seek to access services outside the I2P network, more outproxies may have to be provided. Apart from that, more clients joining the network provides a lot of benefits:

1) The amount of potential fast tier peers will most likely increase and therefore tunnels with less latency and more bandwidth may be the consequence.
2) The amount of cover traffic in the network will most likely increase and therefore provide a stronger anonymity.
3) With more clients using the network, it is very likely a greater range of services will be provided.

Centralization: In Tor, the network is not fully distributed as it is in I2P. The information about the relay nodes and the hidden services are provided by (currently 9) authorized directories which are placed in US and Europe. These authorized directories keep track of changes in the network and distribute this information, therefore if all of them collude the anonymity is endangered. However, in I2P such centralization doesn't exist. Each participating relay locally maintains a list of all known relays.

Routing and Node Selection: Both Tor and I2P run specific node selection algorithms to improve performance and protect against adversaries. While Tor distinguishes between entry, exit and intermediate nodes, I2P has none such. In case of I2P, each peer selected for a specific tunnel may either be the first, the last or an intermediate peer. To be able to select intermediate and exit routers, Tor's directory servers use active bandwidth probing to measure and record the bandwidth each OR is able to provide. This generates non-data message traffic. Also Tor has to rely on self-advertised bandwidth values if no probing data is available for this specific OR. This may lead to misclassification or may potentially be used by an adversary to classify his OR as an entry guard. With the exception of the entry guard, which is chosen from a small set of well-known ORs with long uptime, all other ORs in Tor are chosen with a probability proportional to its bandwidth [3]. This means only bandwidth and capacity are considered while other attributes like the actual location of the ORs are ignored. This may lead to high latencies when ORs are chosen that are, for example, located on different continents. In case of bandwidth, this way of selecting nodes may be optimal, but when taking into consideration, that the latency is an important point when browsing websites, this may not seem to be the optimal way. The current load of the network is also not considered; therefore, existing resources may not be optimally used [3].

I2P clients on the other hand solely rely on previously monitored performance values and the current state of the network. No active bandwidth probing is used. The I2P node selection algorithm is also able to react very fast to failing peers and other changes in the network topology. This behavior of quickly reacting to failing nodes also holds a security problem. As described by Herrmann and Grothoff [20], a selective DoS attack targeting the current fast tier peers may give an adversary the possibility to inject his own corrupted peers into the fast tier. Due to the short lifetime of tunnels, some I2P users will most likely use one or more corrupted peers to build their tunnels. This of course may lower the grade of anonymity provided for this particular I2P users. The location of I2P peers is also not considered when categorizing them into tiers, therefore, similar to Tor, high latencies may be the result. Last but not least, newly joined I2P peers may have insufficient or outdated peer statistic and network informations to select optimal tunnel peers.

Avoiding Congestion: Tor uses circuit switching, whereas I2P uses packet switching, hence, Tor has often to cope with high congestion leading to high latency [26]. Whereas in I2P, the packet switching leads to some implicit load balancing and helps to avoid congestion and service interruptions. This is specifically important for large file transfers and therefore I2P is more suitable for such purposes.

Usage: I2P offers several applications and is rather designed for communication within the I2P network, in particular because it has few out-proxies. Whereas Tor is rather designed for routing traffic outside the network and has in comparison to I2P more exit nodes. In addition as mentioned in earlier in this section Tor's performance is better for visiting web pages than I2P, which makes Tor a better choice for surfing. However, for downloading I2P shows better results hence for applications such as file sharing I2P is more suitable.

Attacks: There are essentially two main classes of attacks that target the Tor network: *Traffic analysis attacks* [27], [28] and *DoS attacks*. Attacks on Tor have been commonly reviewed in the literature [29][1][30]. Grahn et al. give a review on general anonymous communications, which includes Tor and I2P [31]. Zantout et al. describe I2P and the known attacks on I2P where the main attacks are classified as DoS attacks, Partitioning attacks, and Intersection attacks [32]. Here, we review some new attacks for both Tor and I2P. For our review the main goal of the adversary is to identify peers (in the case of I2P) or de-anonymizing users (in the case of Tor).

Recently, two DoS attacks have been proposed on Tor: Johnson et al. propose the Sniper attack, which exploits the reliable data transport in Tor by consuming a large amount of memory [33], and Barbera et al. propose the CellFlood attack [34], which exploits the circuit construction process in Tor by flooding the router with circuit construction requests. Both attacks exploit technical vulnerabilities of Tor. Another known DoS attack on Tor, proposed in 2007 by Borisov et al., is the selective DoS attack [35], which rather describes the method for selecting nodes for the DoS attack and is not proposing technical measures for performing the attack. The main goal of DoS attacks against Tor is to either force users to choose malicious routers which in turn reduces the smaller user base and weakens anonymity [36]. More recently, website fingerprinting attacks on Tor have been proposed by Wang et al. which seek to deanonymize users by matching packet quantities and sizes of received packets [37].

Herrmann et al. proposed an attack on I2P uses a sort of selective DoS attack and exploits the node selection bias towards nodes with good performance in order to de-anonymize peers that are hosting Eepsites [20]. Crenshaw investigated de-anonymization attacks on I2P connections by analyzing the data that is leaked by applications that are run using I2P [38]. More recently, Egger et al. proposed some practical attacks on I2P, where the attacker tries to break the anonymity of users by using DoS and Sybil attacks [39] as part of her attack scenario [40].

V. Conclusion

In this paper, two state of the art Onion Routing based low latency anonymous communication systems were presented and compared. While Tor is the at the moment most popular and most used system, I2P is a fast growing competitor. Both systems are constantly being updated to improve performance and provide better anonymity while protecting against adversaries. Tor, due to the fact of it's greater awareness in the academic community, was already able to solve problems that I2P will sooner or later have to face. Tor also benefits from a large number of formal studies of its anonymity, resistance to attacks, and performance. As pointed out before, the key difference of both networks is the way they set up and use their virtual connections, in terms of node selection and client's node participation. Another important difference is that while Tor was designed for exit traffic, I2P seeks to provide services inside the network to provide stronger anonymity for service provider and users.

Overall, this comparison shows that it highly depends on the field of application to determine which system delivers better results in terms of performance and anonymity. When browsing the public web, Tor undoubtedly delivers better performance, while I2P is almost unusable. On the other hand, I2P provides a stronger anonymity and better performance compared to Tor when interacting with services or users inside the network. In the end, it is always a trade-off between performance and anonymity, no matter which system is used.

References

[1] M. Edman and B. Yener, "On anonymity in an electronic society: A survey of anonymous communication systems," ACM Computing Surveys (CSUR), vol. 42, no. 1, 2009, pp. 5:1–5:35.

[2] "The Tor project," https://www.torproject.org/, accessed: 11/07/2014.

[3] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document, Tech. Rep., 2004.

[4] "The Invisible Internet Project project," https://geti2p.net/en/, accessed: 11/07/2014.

[5] "The FreeNET project," https://freenetproject.org/, accessed: 11/07/2014.

[6] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability, 2001, pp. 46–66.

[7] "Project: AN.ON- anonymity.online," http://anon.inf.tu-dresden.de/index_en.html, accessed: 11/07/2014.

[8] O. Berthold, H. Federrath, and S. Köpsell, "Web MIXes: A system for anonymous and unobservable Internet access," in Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 115–129.

[9] "The GNUnet project," https://gnunet.org/, accessed: 11/07/2014.

[10] G. Danezis and C. Diaz, "A survey of anonymous communication channels," Computer Communications, vol. 33, 2008.

[11] J. Ren and J. Wu, "Survey on anonymous communications in computer networks," Computer Communications, vol. 33, no. 4, 2010, pp. 420–431.

[12] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the Tor network," in Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008), N. Borisov and I. Goldberg, Eds. Leuven, Belgium: Springer, July 2008, pp. 63–76.

[13] R. Snader and N. Borisov, "A tune-up for Tor: Improving security and performance in the Tor network," in Proceedings of the Network and Distributed Security Symposium - NDSS '08. Internet Society, February 2008.

[14] A. Panchenko, F. Lanze, and T. Engel, "Improving performance and anonymity in the Tor network," in Proceedings of the 31st IEEE International Performance Computing and Communications Conference (IPCCC 2012), December 2012, pp. 1–10.

[15] J. Timpanaro, I. Chrisment, and O. Festor, "I2P's usage characterization," Traffic Monitoring and Analysis, 2012, pp. 48–51.

[16] J. Timpanaro, C. Isabelle, F. Olivier et al., "Monitoring the I2P network," 2011.

[17] M. Ehlert, "I2p usability vs. Tor usability a bandwidth and latency comparison," Seminar Report, Humboldt University of Berlin, November 2011.

[18] "I2p...the *other* anonymous network," http://sempersecurus.blogspot.de/2011/06/i2pthe-other-anonymous-network_18.html, accessed: 08/07/2014.

[19] "stats.i2p - the home for NetDB statistics," http://stats.i2p.to/, accessed: 08/01/2013.

[20] M. Herrmann and C. Grothoff, "Privacy-implications of performance-based peer selection by onion-routers: A real-world case study using i2p," in Privacy Enhancing Technologies. Springer, 2011, pp. 155–174.

[21] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in Peer-to-Peer Systems. Springer, 2002, pp. 53–65.

[22] T. E. Y. Iwan Hoogendoorn and J. Soeurt, "Further reducing the anonymity set of web servers hidden within the i2p network," 2011.

[23] zzz (Pseudonym) and L. Schimmer, "Peer profiling and selection in the i2p anonymous network," in Proceedings of PET-CON 2009.1, March 2009, pp. 59–70.

[24] "Tunnel implementation," http://www.i2p2.de/tunnel-alt.html, accessed: 08/07/2014.

[25] "I2P COMPARED TO TOR AND FREENET," http://www.i2p2.de/how_networkcomparisons.html, accessed: 08/07/2014.

[26] R. Dingledine and S. J. Murdoch, "Performance improvements on Tor or, why Tor is slow and what we're going to do about it," The Tor Project, Tech. Rep. 2009-11-001, November 2009.

[27] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in Security and Privacy, 2005 IEEE Symposium on. IEEE, 2005, pp. 183–195.

[28] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against Tor," in Proceedings of the 2007 ACM workshop on Privacy in electronic society. ACM, 2007, pp. 11–20.

[29] N. Danner, S. Defabbia-Kane, D. Krizanc, and M. Liberatore, "Effectiveness and detection of denial-of-service attacks in Tor," ACM Trans. Inf. Syst. Secur., vol. 15, no. 3, Nov. 2012, pp. 11:1–11:25.

[30] T. G. Abbott, K. J. Lai, M. R. Lieberman, and E. C. Price, "Browser-based attacks on Tor," in Proceedings of the 7th International Conference on Privacy Enhancing Technologies, ser. PET'07, 2007, pp. 184–199.

[31] K. J. Grahn, T. Forss, and G. Pulkkis, "Anonymous communication on the internet," in Proceedings of Informing Science & IT Education Conference (InSITE) 2014, December 2014, pp. 103–120.

[32] B. Zantout and R. Haraty, "I2p data communication system," in Proceedings of ICN 2011, The Tenth International Conference on Networks, January 2011, pp. 401–409.

[33] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann, "The sniper attack: Anonymously deanonymizing and disabling the Tor network," in To appear in Proceedings of the 21st Annual Network & Distributed System Security Symposium (NDSS '14). Internet Society, 2014.

[34] M. V. Barbera, V. P. Kemerlis, V. Pappas, and A. Keromytis, "CellFlood: Attacking Tor onion routers on the cheap," in Proceedings of ESORICS 2013, September 2013, pp. 664–681.

[35] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz, "Denial of service or denial of security? How attacks on reliability can compromise anonymity," in Proceedings of CCS 2007, October 2007, pp. 92–102.

[36] R. Dingledine and N. Mathewson, "Anonymity loves company: Usability and the network effect," in Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006), R. Anderson, Ed., June 2006.

[37] T. Wang and I. Goldberg, "Improved website fingerprinting on Tor," in Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society. ACM, 2013, pp. 201–212.

[38] A. Crenshaw, "Darknets and hidden servers: Identifying the true ip/network identity of i2p service hosts," in Proceedings of Black Hat 2011, January 2011.

[39] J. Douceur, "The Sybil Attack," in Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002), March 2002, pp. 251–260.

[40] C. Egger, J. Schlumberger, C. Kruegel, and G. Vigna, "Practical attacks against the I2P network," in Proceedings of the 16th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2013), October 2013, pp. 432–451.