Cryptanalysis of Tor

December 14th 2023

Mount Allison University

Nicholas Nagy

The History of Tor

The idea of Tor was initially aimed to protect American web communications; the idea emerged in the mid-1990s (Tor Project, n.d.). It was a remarkable collaboration between computer scientists Michael G. Reed, David Goldschlag and mathematician Paul Syverson. The first version of Tor was launched on September 20, 2002 becoming publicly available a year later. Tor is also known as The Onion Router, which is open-source software that provides anonymity online (Williams, 2023). It reroutes internet traffic through a global network of volunteer-operated servers, numbering over seven thousand (Wate, 2020). Tor's unique "Onion Routing" principle encrypts messages in multiple layers, much like the layers of an onion; it operates by directing your data through a series of three randomly selected servers, known as relays, within its network. The final relay in this sequence, referred to as the "exit relay", then releases your data onto the public internet. Going forward I will be referring to relays as nodes since I will be doing graphical analysis of the network.

Understanding The Structure of Tor

The network uses three nodes to connect a client to the desired destination; each node using the AES encryption standard to encrypt and decrypt data (Tor Project, 2023). Specifically, Tor uses the TLS ciphersuite "TLS_DHE_RSA_WITH_AES_128_CBC_SHA" for network connections and AES in counter mode to encrypt/decrypt data. The key exchange Tor uses is the x25519 key agreement scheme which is based on elliptic curve cryptography: we will not have time to discuss the key agreement scheme sadly. To summarize, the data transmitted through Tor

is encrypted and decrypted by a symmetric encryption algorithm which will be discussed in more detail later on; it is widely used across the globe due to its robustness and efficiency.

The general structure follows: devices connecting to the network will encrypt the message three times using three separate 128 bit keys; each encryption will run consecutively (Arora, Pankaj, & Banerjee, 2018). When data is encrypted a header section will be created and attached to the encrypted data: the header serves as a repository for meta-information, encompassing details such as the destination address (Wate, 2020). When you initiate a request, the guard node in the network receives your data and removes the first layer of encryption, similar to peeling an onion. This guard server is aware of your IP address but is oblivious to the site you are attempting to access. It only knows the address of the subsequent server from the header information. The following node is then unaware of your IP address or the site you are trying to visit; its sole function is to strip away a layer of encryption and forward the data to the last node. Upon reaching the final node, it eliminates the last layer of encryption and directs your web request to its intended destination. On the other hand, the response from the website will then get encrypted at each node with the same key and decrypted on the device.

Since this network contains at least seven thousand nodes; a guard node, middle node and final node are chosen at random creating a route (Arora, Pankaj, & Banerjee, 2018). Each route in the network is maintained for a brief period of no more than 10 minutes once a connection is established. This route is then discarded, making it impossible to trace back. Tor operates as a network of proxy servers, with each node only having enough information to decrypt the data packet and forward it to the next node; although the traffic between the exit node and the destination is not necessarily encrypted. Tor also allows for the hosting of websites through hidden services (Tor Project, n.d.). In contrast to conventional services, hidden services offer

bidirectional anonymity since the information is stored in the Tor network itself; this greatly increases the security of the network. Just like any service accessed over Tor, the server is unaware of the client's IP address; however the unique aspect of hidden services is that the client is also oblivious to the server's IP address. This dual-sided privacy protection offers the highest level of confidentiality. These services provide no identifiable information about the server; such being its location, ownership, source of traffic, or data flow direction.

The configuration of the network may undergo alterations when a Tor bridge is incorporated: they serve as alternate access points into the Tor network, functioning as replacements for the standard Tor guards (Tor Project, n.d.). Bridges are operated by volunteers and are not publicly listed, which makes it more difficult for adversaries to detect them: this aids in masking your usage of Tor. While a bridge can make it more challenging for your Internet Service Provider to ascertain whether you are connecting to Tor; it does not make it impossible. This can be especially beneficial if your access to Tor is restricted or if your use of Tor could raise suspicions for those monitoring your internet connection.

As data exits the Tor network via the exit node, it transitions into the open internet (Arora, Pankaj, & Banerjee, 2018). This juncture is where your data might be exposed to potential risks: while most Tor exit nodes are secure, a few can pose problems. Your internet traffic is susceptible to surveillance; the exit node, the point at which your traffic departs the Tor network and enters the open Internet, can be observed. Consequently, it's strongly recommended to avoid transmitting unencrypted data over the Tor network as someone might intercept your confidential information during its transit to the final destination. While pinpointing the source of the data could prove to be a formidable task, any user identification leaving the exit node could be readily accessed by anyone; thereby negating the purpose of using TOR for anonymity.

The Disadvantages of Tor

Building on the previous topic, we will now transition into examining the potential downsides of using Tor; one being the security of your data when utilizing the Tor network is dependent on the website you are accessing (Williams, 2023). If the website supports HTTPS then your connection is encrypted; this guarantees the safety of your data even as it departs the Tor network and navigates through the internet. The Tor Browser itself is designed to enable users to access the Tor network, and fortifies this security with an HTTPS-Only Mode. This mode insists on the use of HTTPS encryption for all websites that are compatible. Consequently, if a website offers both HTTP and HTTPS, the Tor Browser will then default to the more secure HTTPS connection. It is crucial to remember that not all websites are equipped with HTTPS: in instances where a website only supports HTTP, your connection is devoid of encryption and could potentially be exposed to third parties.

We know the Tor network places a high emphasis on privacy and anonymity but a significant drawback of using it is the potential for slower speeds (AlSabah & Goldberg, 2015). The slower speeds are caused by multiple layers of encryption, routing across geographically dispersed nodes, and network congestion. Methods such as random routing and intentional delays, all of which are used to thwart traffic analysis can also add to the perceived sluggishness. It is crucial for users to understand these compromises but nonetheless it is worth noting that ongoing enhancements are being implemented to mitigate these issues related to speed.

The performance of connections in the Tor network can be influenced by several factors: one such factor is the broadband speed of the individuals operating the node or bridge (AlSabah & Goldberg, 2015). In addition to this, hardware constraints could also lead to subpar performance of a specific node. It is important to note that these bridges or nodes are typically

short-lived; a consequence to this is if a file transfer or download is interrupted, reestablishing the connection might take some time.

In terms of managing network traffic we know Tor employs flow control but lacks a mechanism for congestion control. Flow control is a method that adjusts the data transmission rate between two nodes; it is designed to prevent a situation where a fast sender overwhelms a slower receiver (GeeksforGeeks, 2023). By ensuring that the sender does not transmit data too rapidly for the receiver to handle, it helps avoid buffer overflow. This overflow can result in dropped packets and degrading network performance. The major downside of flow control is that it can introduce delays in data transmission due to its regulation of data flow; it is also less effective in congested networks where multiple sources cause congestion. On the other hand, congestion control is a technique designed to avert network congestion. Network congestion occurs when an excess amount of data is transmitted over a network, causing it to become overloaded; this can lead to dropped packets and poor network performance. Congestion control mitigates network congestion by controlling the data transmission rate from the sender to the receiver promoting fair allocation of network resources by managing the data flow rate for all sources. Sadly, congestion control can also introduce delays in data transmission and might result in underutilization of network resources if the congestion is not severe.

The absence of congestion control means that the nodes within a route are unable to respond to or protect themselves from congestion occurring at the overlay layer (AlSabah & Goldberg, 2015). To understand the overlay network, we will be visualizing the Tor network as a graph composed of nodes and edges. Each device within the Tor network represents a node, and each connection between devices represents an edge. This forms the physical network graph; the overlay network can be seen as a secondary graph superimposed on the primary one. The nodes

remain the same, representing the devices but the edges in the overlay network symbolize virtual connections between these devices, which can encompass multiple physical connections. Thus, the overlay network is essentially a virtual graph overlaid on top of the physical network graph allowing for the creation of new indirect paths between nodes. To address this issue, AlSabah proposed the N23 algorithm: originally designed for Asynchronous Transfer Mode networks, this congestion control algorithm offers several benefits. It helps reduce route queues in nodes, enables them to signal congestion via back pressure and allows for a swift response to changes in available bandwidth.

Advantages of Using Tor

Tor's primary benefit is the anonymity it provides as it conceals your IP address and other system details; this enables you to browse the internet without disclosing your identity (Arora, Pankaj, & Banerjee, 2018). In addition, Tor strengthens network security with its triple encryption system, rendering any attempts to crack the cipher on the client side virtually unachievable. Another advantage of Tor is its ability to access the deep web and blocked websites: it also supports .onion services, which are linked to the deep web and can only be accessed via this browser.

Creating an onion service can also only be done using Tor. In its conception, an onion service begins by creating a descriptor, which is a document that contains important information about the service; this includes a list of introduction points. Introduction points are Tor nodes that the service has selected to help establish connections with clients. To ensure the authenticity of the descriptor, the service signs it using its private key: this is a unique cryptographic key that only the service knows, and it allows anyone who has the service's public key to verify that the descriptor really came from the service. The signed descriptor is then uploaded to a distributed

hash table, which is a system used by the Tor network to store and retrieve descriptors. The hash table is distributed across many Tor nodes which allows any client who knows the service's onion address to download the service's descriptor. The descriptor is sent over a Tor route that maintains anonymity; meaning that even the Tor nodes that store the descriptor in the hash table don't know the location of the service. So, while the onion service uses the Tor network to publish its descriptor and to communicate with clients, the service itself is hosted on a server that could be anywhere on the internet, outside of the Tor network. When a client aims to connect to an onion service, they do not establish a direct connection (Williams, 2023). Instead, a route forms through the network where you are hopping from one node to the next until it reaches the desired onion service. The client selects the guard node as the starting point, then moves to the second node, and continues this process until it arrives at the exit node, which provides the connection to the onion service.

Understanding The TLS Protocol

To comprehend the ciphersuite utilized by Tor, it is essential to first gain an understanding of the Transport Layer Security protocol. It is also known as TLS and it is a cryptographic protocol that ensures secure data transmission over the Internet (Cloudflare, n.d.). The process initiates with a TLS handshake, during which the client and server establish the TLS version; afterwards they both select the cipher suite to utilize and authenticate each other's identities using digital certificates. This is followed by a key exchange where a random string of bytes is generated, which is subsequently encrypted using the server's public key beginning obtained from the server's SSL certificate; this is then transmitted to the server. This string of bytes, also known as the pre-master secret, is then used to create symmetric encryption keys. These keys are designed to be obtained only by the client and server, thereby preventing

eavesdropping. Upon completion of the handshake and establishment of symmetric keys, data transfer commences in which the client and server begin to encrypt and decrypt the data they exchange using the agreed-upon symmetric encryption algorithm. Finally, each piece of data is transmitted with a Message Authentication Code, also known as MAC; this enables the recipient to verify the integrity of the data and confirm that it has not been tampered with during transmission. In essence, TLS facilitates a secure connection by authenticating the client and server; later setting up encryption parameters, and finally maintaining data integrity.

The Ciphersuite at Each Node

The cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA plays a pivotal role in the TLS protocol, which is extensively employed to safeguard communication between two devices (Stack Exchange, 2021). It is also the bare minimum requirement to run on the Tor Network; if a better ciphersuite is available then that will be used for communication instead. This suite is an amalgamation of various cryptographic algorithms, each of which contributes uniquely to the process of secure communication.

The process begins with the Diffie-Hellman Ephemeral, a secure variant of the original Diffie-Hellman protocol; it is designed to enhance security by generating a unique set of key pairs each time the protocol is executed (upGrad, 2020). This feature ensures perfect forward secrecy; this implies that even if a key is compromised, it cannot possibly decrypt past messages. The DHE key exchange process involves both parties publicly agreeing on a generator and a prime modulo. Following this, each party chooses a private integer and calculates a corresponding public key; each of these public keys are then exchanged, allowing each party to compute the shared secret. The term ephemeral in DHE signifies that these private integers and

resulting public keys are freshly generated for each session; this enhances the security of the communication.

Following DHE, the RSA algorithm comes into play for server authentication (Stack Exchange, 2021). The server's digital certificate serves as a form of identity verification, and must include a public key derived from the RSA algorithm. This public key is utilized by the client to encrypt the pre-master secret during the TLS handshake. In addition, the server possesses a corresponding private key, which is securely kept hidden. This private key has two main functions: it decrypts the pre-master secret that the client sends, and it signs the generator and prime number; this being a critical step in the DHE key exchange.

Next, data encryption is performed using the Advanced Encryption Standard with a 128-bit key size with Cipher Block Chaining mode. The data encrypted in this case is not actually the plaintext but the ciphertext from AES in counter mode. AES with CBC mode is only there for the transfer of data between nodes and will be decrypted once the node destination is reached. Cipher Block Chaining is a method used in block ciphers to ensure data confidentiality: in this mode each plaintext block is combined with the previous ciphertext block using an XOR operation before it is encrypted (DevX, 2023). This process guarantees that even if two blocks of plaintext are identical, they will not result in the same ciphertext after encryption. It is important to note that CBC mode has been associated with several vulnerabilities; some including BEAST, Lucky-Thirteen, and POODLE attacks, which will be discussed later. Finally, a Secure Hash Algorithm is used to verify data integrity; in this case the standard is SHA-256. Despite its widespread use, this cipher suite may not be the most secure option due to potential vulnerabilities associated with CBC mode. A more secure alternative could be a suite that

employs AES in Counter Mode; which is the same as Tor uses for standard data encryption, and known for its good performance and decent security.

Vulnerabilities of the Ciphersuite

As previously mentioned, a node in the network utilizes the ciphersuite mentioned in the worst case scenario, which generally provides robust security. The use of CBC mode can potentially open up avenues for specific attacks under certain conditions; grasping the security aspects of connections between nodes can pave the way to comprehending the security of the entire Tor network. In this context, we will delve into potential vulnerabilities of this cipher suite, including BEAST, Lucky-Thirteen, and POODLE. Additionally, we will briefly touch on recently identified attacks, namely Zombie POODLE and GOLDENDOODLE, which exploit the aforementioned vulnerabilities.

The BEAST attack, short for Browser Exploit Against SSL/TLS, is a type of cryptographic timing attack (Kiprin, 2021). It targets a vulnerability found in the TLS 1.0 and older SSL protocols, specifically when using cipher block chaining mode encryption. This attack allows for the interception and decryption of HTTPS client-server sessions; in this case the attacker would not be able to see the data or the location of the next node in the route since it would need to breach the second cipher which is AES in counter mode.

The next attack is another cryptographic timing attack, known as the Lucky-Thirteen attack (Kiprin, 2021). This attack can be used against implementations of the TLS and Datagram Transport Layer Security protocols that use the Cipher Block Chaining mode of operation. It can be considered a type of man-in-the-middle attack: the vulnerability that enables the Lucky-Thirteen attack affects the TLS 1.1 and 1.2 and DTLS 1.0 or 1.2 implementations, as well as previous versions such as SSL 3.0 and TLS 1.0.

Finally, the POODLE attack, or Padding Oracle on Downgraded Legacy Encryption, which exploits a vulnerability in the SSL 3.0 protocol (Acunetix, 2020). This vulnerability allows an attacker to eavesdrop on communication encrypted using SSLv3; it has been addressed in the TLS, the successor to SSL. The POODLE vulnerability allows an attacker to steal confidential data transmitted over the network, such as passwords or session cookies, enabling them to impersonate the user.

The vulnerabilities known as Zombie POODLE and GOLDENDOODLE bear similarities to previously discussed attacks, but they have been identified to function with more recent versions of TLS (Qualys, 2019). As of now, TLS 1.3 remains secure with no known exploits; to exploit any of the mentioned vulnerabilities, one would need to deliberately utilize an outdated version of TLS. Tor supports older versions of TLS, which potentially allows an attacker to alter the TLS version in use. The primary challenge after breaching the cipher suite is the triple-layer encryption employed by Tor, specifically AES in counter mode. To determine the data's destination, one would need to intercept the data, crack the cipher suite, and then crack the first layer of AES in counter mode: this sequence of tasks is considerably challenging.

Traffic Analysis

Having examined the potential vulnerabilities of the ciphersuite, we will now shift our focus to the vulnerabilities affecting the network as a whole. We will now focus on how the Tor network could potentially be compromised through traffic analysis. Traffic analysis involves studying communication patterns to gather intelligence about a system or its users, such as users on the Tor network in our case (National Institute of Standards and Technology, n.d.). This method does not require the analysis of the actual communication content, which might be impossible to decipher; when applied to Tor, a basic method of traffic analysis involves

monitoring network traffic prior to the guard node and following the exit node. However, this approach requires an informed hypothesis regarding the user's intended actions.

Tor's low-latency characteristic limits its ability to fully guard against traffic analysis attacks (Basyoni, Fetais, Erbad, Mohamed, & Guizani, 2020). The term "low-latency" in the context of Tor, signifies its design objective to reduce data transmission delays across the network. While we have established that Tor operates at a slower pace compared to conventional search engines, it compensates for this by striving to minimize delays as much as possible. These attacks seek to de-anonymize Tor users by watching the traffic that enters, exits, or traverses the Tor network. The effectiveness of these attacks hinges on the accuracy of the adversary's information: the more comprehensive the adversary's network coverage, the more likely the monitored traffic will be accurate. Recent advancements in traffic-analysis techniques have enabled adversaries with only a partial network view to deduce the nodes used to relay anonymous streams which significantly diminishes the anonymity Tor provides.

An example technique is one that utilizes machine learning classifiers; in particular stacking ensemble learning is employed to examine and classify traffic (Schüler & Petrik, 2023). This method initiates with the gathering of data related to network traffic. Following this, three fundamental learning techniques are applied to analyze the data; each of these techniques formulates its own predictions about the traffic. These predictions are subsequently fed into the stacking ensemble learning system in which the system then makes a conclusive decision about the nature of the traffic; categorizing it as either benign or malignant similarly to how tumors are identified in cancer patients. Despite its effectiveness, it's important to note that the implementation of this method demands substantial resources and expertise.

Attacking the Tor Protocol

One could contend that the current methods of traffic analysis attacks lack the effectiveness, robustness, and stealth required to significantly undermine the anonymity provided by the Tor network. While there are still potential vulnerabilities in a traffic analysis of the Tor network; the possibility of breaching a user's anonymity is still very unlikely. Another method of attacking the Tor network is by attacking the protocol itself; the most vulnerable being the Transmission Control Protocol aka TCP which is used to facilitate the transmission of data as a continuous stream of bytes.

In such attacks, a rogue guard node could potentially tamper with the cells of a TCP stream originating from a sender; in this case a cell represents the fundamental unit of data transmission in Tor (Fu & Ling, 2009). This tampering could include duplicating, altering, inserting, or deleting cells; the attacker merely needs to manipulate a single cell to verify the communication link between the sender and receiver.

These altered cells then pass through the middle node and reach the exit nodes along the route (Fu & Ling, 2009). Tor uses the counter mode of the Advanced Encryption Standard, aka AES-CTR, for the encryption and decryption of these cells at the nodes; while the connections between nodes use the TLS cipher suite discussed above. AES in Counter mode is a widely used method that converts a block cipher into a stream cipher by creating a keystream and using an encrypted counter value. This keystream is then XORed with the plaintext to generate the ciphertext. The same keystream can be utilized for both encryption and decryption. One of the distinctive features of CTR mode is that all steps can be executed in parallel. This is possible because the counter value is incremented for each block, ensuring that every block is encrypted with a unique keystream. This essentially aims to ensure that no plaintext blocks are ever

repeated for the same key. CTR mode bears similarity to Output Feedback mode, but the main difference lies in the generation of these pad vectors. In CTR mode, the pad vectors are generated by encrypting a counter value.

The manipulated cells can disrupt the normal counter at the exit onion routers, leading to cell recognition errors during decryption (Fu & Ling, 2009). These errors are a unique characteristic of the protocol-level attacks being investigated. If an accomplice of the attacker also controls the exit nodes and can identify these cell recognition errors, they can confirm the communication relationship between the sender and receiver. This poses a significant threat to the anonymity that Tor provides, as it allows the attacker to confirm anonymous communication relationships quickly and accurately. This constitutes a significant risk to Tor's anonymity, as it enables the attacker to swiftly and accurately confirm anonymous communication links. Such attacks are deemed effective and present a substantial threat to Tor's anonymity. They differ from existing attacks and can breach Tor's anonymity by gaining control over both the entry and exit nodes in the route. Similar to a traffic analysis this requires a decent amount of network control. While we are on the discussion of the Tor protocol I wanted to mention that AES-CTR is supposed to be replaced with AEAD+SPRP to defend against the aforementioned attack discussed above (The Tor Project, 2023). There have also already been other implementations that have mitigated the risk of such an attack.

Sybil Attacks

A different type of attack that can be done to the network is known as a Sybil attack; it is when the attacker creates multiple nodes on the network (GeeksforGeeks, n.d.). These nodes then appear to be real unique nodes to outside observers but allow the attacker to have a disproportionately large influence on the network; they can then use these Sybil nodes to monitor

a significant fraction of the network's traffic. This leads to a potential breach of the privacy and anonymity that Tor aims to provide. There are two types of Sybil attacks: direct and indirect in which a direct attack, the honest nodes are influenced directly by the Sybil nodes. While on the other hand, an indirect attack is where the honest nodes are attacked by a node which communicates directly with the Sybil nodes. This middle node is then compromised as it's under malicious influence of Sybil nodes. Sybil attacks on the Tor network have occurred in reality, with significant instances including a combined Sybil and traffic confirmation attack that targeted the Tor network for several months in 2014, as well as the Bitcoin address rewrite attacks in 2020. These attacks pose a substantial threat, however steps are being implemented to identify and counteract such attacks; thereby safeguarding the privacy and anonymity of users on the Tor network.

ShorTor: An Upgraded Version of Tor

ShorTor is a protocol that employs multi-hop overlay routing to decrease latency in the Tor network; in addition to this it ensures that security and anonymity are not compromised. Multi-hop overlay routing is a method that incorporates intermediate waypoints into the connection between a client and a server; its aim is to bypass slow or congested default routes (Hogan et al., 2022). In terms of security, ShorTor upholds the anonymity guarantees of Tor by selecting nodes based on the adjacent route nodes and utilizing data races to determine the quickest path. ShorTor also utilizes the Ting Protocol, the Ting protocol serves as a key measurement technique: it operates by establishing ephemeral Tor routes between a duo of observer nodes and a pair of target nodes. The protocol then measures the time taken for a round trip through this route. The primary function of this protocol is to gather latencies between pairs of Tor nodes, a critical aspect for the operation of the ShorTor protocol.

When evaluating its performance, it has been observed that ShorTor reduces the latency for the 99th percentile of node pairs in Tor by 148 ms and for Tor routes by 122 ms (Hogan et al., 2022). These findings are based on measurements taken from real-world scenarios: ShorTor was evaluated using pairwise latencies among the most frequently used 1,000 Tor nodes. The results indicated that ShorTor is capable of reducing the round-trip time for 25.4% of node pairs by a minimum of 50% and it also has the ability to eliminate high tail latencies. On the security front, the AnoA framework was employed to examine the effect of ShorTor on anonymity. The findings suggested that while ShorTor does not influence the anonymity of baseline Tor, it could potentially compromise the anonymity of location-aware path selection schemes.

The AnoA framework serves as a method for defining, analyzing, and quantifying the anonymity attributes of anonymous communication protocols like Tor (Backes et al., 2013). These anonymity attributes, which include sender anonymity, recipient anonymity, sender unlinkability, and relationship anonymity, safeguard the identity and privacy of protocol users. The framework employs a unique adaptation of differential privacy; a mathematical metric that measures the amount of information leaked by a randomized algorithm. Differential privacy assesses privacy loss by comparing the probability of an output given two distinct inputs. The AnoA framework modifies this concept to compare the likelihood of an adversary's guess given two different messages sent via an anonymous communication protocol.

The AnoA framework enables the provision of anonymity assurances for various anonymity attributes through explicit anonymity functions that formalize each attribute (Backes et al., 2013). These functions encapsulate the advantage and confidence of an adversary in breaching user anonymity; the framework also delineates adversarial capabilities using a straightforward wrapper-construction known as adversary classes. These classes specify the kind

and quantity of information an adversary can access and utilize in their attacks. The AnoA framework is compatible with the Universal Composability framework, a commonly used framework for validating the security of cryptographic protocols. The AnoA framework can also be applied to real-world systems like Tor to analyze their anonymity attributes against passive attackers.

Conclusions

In conclusion, our exploration has encompassed the cipher suite employed by Tor nodes, the TLS protocol, and the potential threats that could leverage the CBC mode of encryption; we also discussed attacks on Tor's protocol. We've delved into Traffic Analysis and Sybil Attacks, both of which are strategies that adversaries could employ to de-anonymize Tor users by observing or controlling a significant portion of the network's traffic. Lastly, we've considered an improved alternative known as ShorTor, a protocol designed to decrease latency in the Tor network. While it doesn't enhance Tor's security, it does facilitate a quicker and more responsive user experience. Additionally, we've discussed AnoA, a framework that carefully examines and quantifies the anonymity characteristics of anonymous communication protocols.

References

ABOUT TOR BROWSER. (n.d.). Tor Browser Manual. Retrieved December 10, 2023, from https://tb-manual.torproject.org/about/

Almomani, A. (2023). Darknet traffic analysis, and classification system based on modified stacking ensemble learning algorithms. *Information Systems and E-Business Management*, 1–32. https://doi.org/10.1007/s10257-023-00626-2

AlSabah, M., & Goldberg, I. (n.d.). Performance and security improvements for tor: A survey. *Cryptology ePrint Archive*.

Arora, S., Pankaj, A., & Banerjee, P. (n.d.). *Anonymity and anonymous connections using TOR*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3318405

Bernaschi, M., Celestini, A., Cianfriglia, M., Guarino, S., Lombardi, F., & Mastrostefano, E. (2022). Onion under Microscope: An in-depth analysis of the Tor Web. *World Wide Web*, *25*(3), 1287–1313. https://doi.org/10.1007/s11280-022-01044-z

Cipher block chaining. (2023, October 18). DevX. https://www.devx.com/terms/cipher-block-chaining/

Difference between flow control and congestion control. (2019, May 24). *GeeksforGeeks*. https://www.geeksforgeeks.org/difference-between-flow-control-and-congestion-control/

Dutta, N., Jadav, N., Tanwar, S., Sarma, H. K. D., & Pricop, E. (2022, January 1). *TOR—The onion router*. Springer Singapore. https://link.springer.com/chapter/10.1007/978-981-16-6597-4_3

Ghimiray, D. (2022, August 4). The Dark Web Browser: What Is Tor, Is it Safe, and How to Use It. *Avast*. https://www.avast.com/c-tor-dark-web-browser

Hogan, K., Servan-Schreiber, S., Newman, Z., Weintraub, B., Nita-Rotaru, C., & Devadas, S.

(2022, April 9). *ShorTor: Improving tor network latency via multi-hop overlay routing*. arXiv.Org. https://arxiv.org/abs/2204.04489

JRA_TLL. (n.d.). *Is TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 a safe cipher suite to use?* Information Security Stack Exchange. Retrieved December 10, 2023, from https://security.stackexchange.com/questions/72926/is-tls-ecdhe-rsa-with-aes-128-cbc-sha256-a-safe-cipher-suite-to-use

Kane, A. M. (n.d.). Cryptanalysis of a fair anonymity for the tor network. *Cryptology ePrint Archive*.

Kiprin, B. (2021a, April 2). *SSL BEAST attack explained*. Crashtest Security GmbH. https://crashtest-security.com/ssl-beast-attack-tls/

Kiprin, B. (2021b, April 3). *Lucky13 Vulnerability - What is it and how to prevent it*. Crashtest Security GmbH. https://crashtest-security.com/prevent-ssl-lucky13/

Ling, Z., Luo, J., Yu, W., Fu, X., Jia, W., & Zhao, W. (2013). Protocol-level attacks against Tor. *Computer Networks*, *57*(4), 869–886. https://doi.org/10.1016/j.comnet.2012.11.005

Nidecki, T. A. (2020, June 1). *What is the POODLE attack?* Acunetix. https://www.acunetix.com/blog/web-security-zone/what-is-poodle-attack/

ONION SERVICES. (n.d.). Tor Browser Manual. Retrieved December 10, 2023, from https://tb-manual.torproject.org/onion-services/

PGzlan. (2023, June 30). Understanding AES operation modes. *DEV Community*. https://dev.to/0xog_pg/understanding-aes-operation-modes-enhancing-security-and-confidentiality-a9d

proposal for new AEAD+SPRP relay cell protocol (#79) · Issues · The Tor Project / Core / Tor Specifications · GitLab. (n.d.). GitLab.

https://gitlab.torproject.org/tpo/core/torspec/-/issues/79

Ribco, N. (n.d.). *An analysis of the security risks posed by tor browser*. Retrieved December 10, 2023, from https://blog.cyberproof.com/blog/an-analysis-of-the-security-risks-posed-by-tor-browser

Shared Key Encryption - An overview. (n.d.). ScienceDirect Topics. https://www.sciencedirect.com/topics/computer-science/shared-key-encryption

Sharma, R. (n.d.). Top 12 commerce project topics & ideas in 2023 [for freshers]. *upGrad Education*. Retrieved December 10, 2023, from https://www.upgrad.com/blog/diffie-hellman-key-exchange/

Sybil attack. (2019, January 10). *GeeksforGeeks*. https://www.geeksforgeeks.org/sybil-attack/

Tell me about all the keys Tor uses. (n.d.). Support. Retrieved December 10, 2023, from https://support.torproject.org/about/key-management/

The tor project. (n.d.). History. Retrieved December 10, 2023, from https://www.torproject.org/about/history/

traffic analysis - Glossary. (n.d.). CSRC. Retrieved December 10, 2023, from https://csrc.nist.gov/glossary/term/traffic_analysis

Wate, Y. (2019, August 14). What is Tor and Should You Use it? [Explained] - TechPP. *Technology Personalized*. https://techpp.com/2019/08/14/what-is-tor/

What is a bridge? (n.d.). Support. Retrieved December 10, 2023, from https://support.torproject.org/censorship/censorship-7/

What is Transport Layer Security (TLS)? (n.d.). Cloudflare. Retrieved December 10, 2023, from https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/

Williams, M., Drake, N., & Clymo, R. (2023, January 16). What is Tor and how does it work?

TechRadar Pro. https://www.techradar.com/features/what-is-tor-and-how-does-it-work

Zombie POODLE and GOLDENDOODLE vulnerabilities. (2019, April 22). Qualys Security Blog. https://blog.qualys.com/product-tech/2019/04/22/zombie-poodle-and-goldendoodle-vulnerabilities

M. Backes, A. Kate, P. Manoharan, S. Meiser and E. Mohammadi, "AnoA: A Framework for Analyzing Anonymous Communication Protocols," *2013 IEEE 26th Computer Security Foundations Symposium*, New Orleans, LA, USA, 2013, pp. 163-178, doi: 10.1109/CSF.2013.18.

L. Basyoni, N. Fetais, A. Erbad, A. Mohamed and M. Guizani, "Traffic Analysis Attacks on Tor: A Survey," 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), Doha, Qatar, 2020, pp. 183-188, doi: 10.1109/ICIoT48696.2020.9089497.