# The Tor Network - Strengths and Weaknesses

Computer Security 1 - CS5460

LaTeX

Jesse Victors

November 2013

## I. Introduction

The Tor network is a second-generation onion routing system that aims to provide anonymity, privacy, and Internet censorship protection to its users. Tour routes encrypted TCP traffic through a worldwide network of over four thousand relays run by volunteers across the world. Tor's encryption, authentication, and routing protocols are designed to make it infeasible for any adversary to identify an end user or reveal their traffic. Throughout its history, various organizations and governments have attempted to block, tap, or crack the Tor network. It was recently revealed that the US National Security Agency has attempted to penetrate the Tor network. Tor is also currently undergoing a transition from RSA-based to elliptic-curve-based TLS. In light of these attacks, Tor's popularity, and the protocol transition, a question that must be asked both by its users and by outsiders is: how secure is Tor? How does it work, what does it provide, and what are its weaknesses? In this paper, I attempt to address these questions.

## II. Design

Tor provides an anonymity and privacy layer by relaying all end-user TCP traffic through a series of *relays* on the Tor network. Typically this route consists of a carefully-constructed three-hop path known as a *circuit*, which changes over time. These nodes in the circuit are referred to as *entry guard*, *middle router*, and the *exit node*, respectively. Only the first node can determine the origin of TCP traffic through Tor, and only the exit node can examine the contents and its destination. Nodes in the middle are unable to determine either. No single node can determine the origin, the contents, and the destination of traffic through the network. Tor's architecture is designed to make it exceptionally difficult for a well-resourced adversary to uncover the identity of the end-user and their network activities, even if nodes are compromised.[6]

### A. Routing

In traditional Internet connections, the client communicates directly with the server. In this model, an eavesdropper can often reveal both the identity of the end user and their activities. Direct encrypted connections do not hide IP headers, which expose source and destination addresses and the size of the payload. In the face of adversaries with sophisticated traffic analysis tools, such information can be very revealing for someone who wishes to hide their activities altogether.

Tor combats this by routing end user traffic through a randomized path through the network of relays. To construct this circuit, the Tor client software first queries a trusted directory server, which return a list of Tor nodes. This is illustrated in Figure 1.

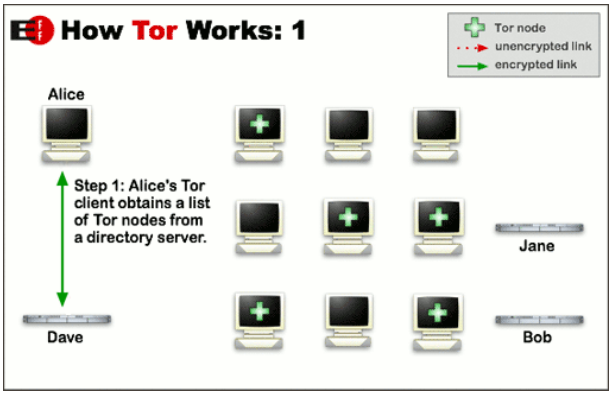


Fig. 1. The user's software first downloads a directory list of Tor relays. This information is later used to construct a circuit through the network.[6][14]

The second step involves choosing three nodes to use and carefully constructing an encrypted path between them. The circuit is extended one hop at a time such that no single relay ever knows the complete path. The client negotiates a separate set of encryption keys for each hop along the circuit to ensure that each hop cannot trace these connections as they pass through, as seen in Figure 2.

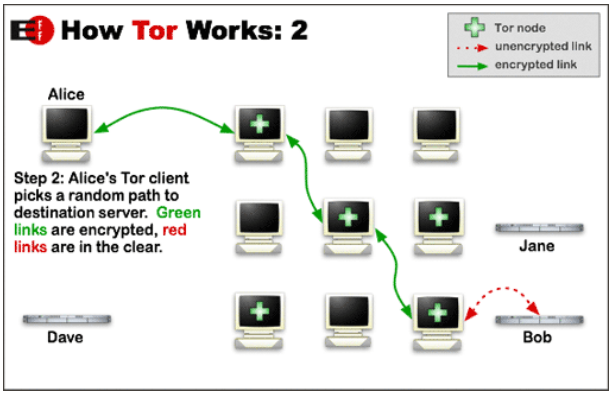


Fig. 2. A Tor circuit is incrementally constructed using layers of encryption. Each node has limited visibility, and no individual node knows the whole circuit.[14]

Following the complete establishment of a circuit, the Tor client software then offers a Secure Sockets (SOCKS) interface which multiplexes TCP traffic through Tor. As each relay sees

no more than one hop in the circuit, neither an eavesdropper nor a compromised relay can use traffic analysis to link the connection's source and destination. Tor further obfuscates user traffic by changing the circuit path every ten minutes,[6] as shown in Figure 3.

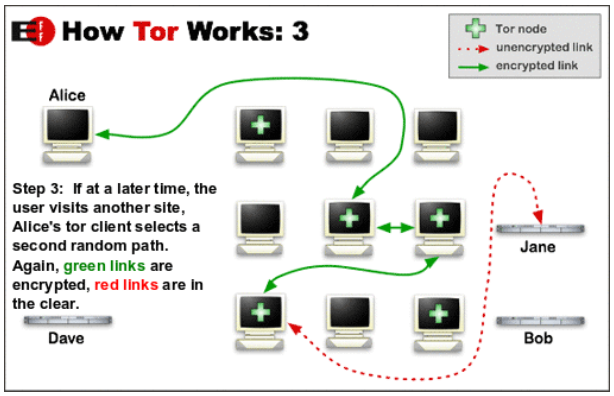


Fig. 3. A Tor circuit is changed periodically, essentially providing a new identity to the end user.[14]

It should be noted that traffic exiting the Tor network is often (but not always) encrypted on its way to the source. An outsider is therefore faced with up to four layers of TLS encryption. Currently, Tor is one of the most secure tools to use against network surveillance, traffic analysis, and to bypassing information censorship.

### B. Encryption

Encryption is a necessary component for privacy within the Tor network. Traffic passing between nodes must be secured from outsiders, and even compromised nodes must not be able to observe the traffic in cleartext. Privacy is also important for traffic outside the network; encrypted connections between the exit node and the target web server must also be achieved. Furthermore, two layers are needed: one for inside the network and one for outside it.

Within the network, nodes talk to each other via the Transport Security Layer (TLS) protocol. When a circuit is built, each pair of nodes within it must first come to an agreement over an encryption key and the specific cipher to use for encryption. There are several mechanisms by which this can be done, all of which use Diffie-Hellman (DH) for key exchange. The most common methods include: TLS_RSA, which rely upon public and private keys generated with RSA; TLS_DHE, ephemeral DH; TLS_ECDH, DH based on elliptic curves, and TLS_ECDHE, ephemeral elliptic curve DH. It should be noted that only TLS_DHE and TLS_ECDHE provide perfect forward secrecy, so it is no surprise that Tor exclusively prefers them. Once encryption keys and the specific cipher (as well as its mode) have been agreed upon, communication of traffic between two nodes can be encrypted and exchanged.

Tor users typically use the Tor Browser Bundle, (TBB) a custom build of Mozilla Firefox with a focus on security and pricacy. The TBB not only provides special handling of client-side scripts such as Javascript, but also offers the HTTPS Everywhere extension, which uses regular expressions to rewrite web requests into ones that use HTTPS. Thus, if the web server is capable of handling SSL or TLS connections, HTTP communications will be encrypted under them. Like any typical browser, the TBB negotiates with the web server for the cipher and keys required for SSL/TSL, except that this negotiation entirely occurs through the Tor circuit. As previously noted, this information, along with subsequent HTTP data, is encrypted between each node in the circuit. During transmission, each node in the circuit decrypts its layer in turn, until the final node (the exit) passes the data to the target server. The TLS/IP connections remain open, so the returned information likewise travels back up the circuit to the end user.

In September 2013, Robert Graham of Errata Security analyzed 22,920 incoming connections to his exit node and found that 89.9% of the circuits agreed to use the Advanced Encryption Standard (AES) block cipher in cipher-block chaining (CBC) mode. This is the most common mode for AES, and allows for parallel decryption of the data blocks. The other 10.1% of the circuits relied upon three rounds of the Data Encryption Standard (triple DES) cipher for encryption. Furthermore, 75.7% of the circuits used elliptic-curve DH, with the remaining 22.3% relying upon traditional RSA DH.[13] Both protocols are discussed below.

*1) RSA:* RSA is an algorithm for public-key cryptography. Its security is based on the infeasibility of factoring the product of two large primes. RSA, like all other public-key cryptography algorithms, relies upon two keys: one public and the other private. The public key may be published and is used for encryption and for verifying digital signatures. The private key is used for decryption, the generation of digital signatures. Thus, through RSA, only the owner of the private key can decrypt incoming messages, and only they can digitally sign outbound messages. This makes RSA extremely powerful for both authenticity and privacy.

To generate the RSA keys, two distinct prime numbers of similar bit-length are chosen at random, $p$ and $q$. $n$ is computed as the product of $p$ and $q$, and it is used as the modulus for both the public and private keys. The bit-length of $n$ is the RSA key length. Then let $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$ where $\varphi$ is Euler's totient function, which counts the number of positive integers less than or equal to $n$ that are relatively prime to $n$. Then $e$ is chosen such that $e$ and $\varphi(n)$ are coprime. $d$ is also chosen such that it is the multiplicative inverse of e modulo $\varphi(n)$. The RSA public key then consists of the modulus $n$ and the exponent $e$. The private key likewise consists of $n$ and the exponent $d$.

A message $M$ is encrypted by first blocks of length less than $n$ using an agreed-upon padding scheme. The ciphertext $c$ is then computed as $m^e \bmod n$, which can be calculated quickly using the method of exponentiation by squaring. The owner of the private key can then recover the plaintext $M$ by computing $c^d \bmod n$, and then reversing the padding scheme. The decryption process is often accelerated by optimizations such as the Chinese Remainder Theorem.

Furthermore, to sign a message, the owner of the private key first uses an agreed-upon hash function to compute a hash value of the message, raises it to the power of $d$ modulo $n$

and then sends that in conjuction with *M*. The recipient, who only has access to the public keys, then raises the signature to the power of *e* modulo *n* and compares the result with his hash of the message using the same hash algorithm. If the two match, then he can prove the authenticity and integrity of the message. These two capabilities make RSA an extremely powerful tool for securing authenticity, integrity, and privacy of communication between two parties.

Tor version 2.3.x and previous rely upon TLS_DHE for the key exchange mechanism. Per the TLS specification, the server generates a Diffie-Hellman public key and sends it to the client. The server also uses its RSA private key to sign everything that it sends to the client during the DH exchange. The client can then confirm the digital signature to verify the authenticity of the communication from the Tor relay.

*2) Elliptic-curve:* Elliptic Curve Cryptography (ECC) is an approach to public-key cryptography in which elliptic curves are used instead of RSA. In contrast to RSA, ECC relies upon the infeasiblity of finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point. This is known as the elliptic curve discrete logarithm problem, or ECDLP.

Elliptic curves over the real number system can be defined through the equation, $y^2 = x^3 + ax + b$ where *a* and *b* are real numbers. This formula describes a plane curve through the Euclidean plane, although elliptic curves do exists in other spaces. However, current cryptographic purposes only work with elliptic curves within the context of the real number systems.

A smaller key size is one of the most significant benefits introduced by ECC. Current NIST recommendations state that a key size of 160 bits for ECC offers a similar level of security to 1024 bits of RSA/DH. Likewise, 224 bits for ECC is analogous to 2048 bits of RSA/DH. It should be noted that the required key sizes for ECC increases significantly slower than the key sizes for RSA and Diffie-Hellman increase at a much faster rate than the required key size for ECC. Thus ECC offers more security per increase in key size than for RSA. The smaller key size also directly correlates to an increase in computational efficiency; as the bit size increases, the speed difference between Diffie-Hellman and ECC grows superlinearly.[9]

Tor 2.4.x introduced TLS_ECDHE, which uses an elliptic-curve variant of the Diffie-Hellman key exchange. Like TLS_DHE, the server uses its RSA key to digitally signs communication to the client. The TLS_ECDHE protocol is believed to be more secure than TLS_DHE.

*3) Symmetric-key:* Once the SSL/TLS handshake completes, both parties use the generated session key and turn to symmetric-key encryption for hiding of data.

Data Encryption Standard (DES) is a block cipher introduced in 1977 by IBM. It was considered insecure in 1998 when the DES Challenges demonstrated that it was feasible to brute-force DES ciphertexts in a matter of hours using custom hardware. This weakness was partially mitigated by the introduction of triple DES (3DES) in the same year, where three rounds of DES were performed using three separate 56-bit keys. However, 3DES was found to be susceptible to meet-in-the-middle, chosen-plaintext, and known-plaintext attacks, so despite using a 168-bit key, 3DES only provides 80 bits of security.[4] 3DES is still used on the web and by older Microsoft products, but has been largely superseded by other algorithms.

Advanced Encryption Standard (AES) is a block cipher introduced in 2001. It supersedes 3DES, and is now used extensively worldwide. AES became the recommended cipher for protecting classified documents for the US government in 2003, and side-channel attacks on specific implementations were the only known vulnerabilities up until mid 2009. To date, the most successful public attacks require $2^1 26.1$ operations to recover a 128-bit key,[7] and related-key attacks can break AES-192 and AES-256 in $2^1 76$ and $2^9 9.5$ operations,[10] respectively.

RC4 is a stream cipher invented in 1987 and sunsequently leaked in 1994. The RC4 algorithm is simple and extremely fast in software, and is now the most widely used software stream cipher, most notably in protocols such as TLS and WEP. Although it is popular, a number of practical attacks and weaknesses have been demonstrated against its Key Scheduling Algorithm and PRNG that call into question RC4's security. In November 2013, Microsoft recommended disabling RC4 from TLS wherever possible.

Robert Graham's analysis of the incoming handshakes to his exit node revealed that almost 90% of the circuits agreed on the AES cipher, while the remainder used 3DES. For communication to web servers, RC4 remained the most popular algorithm.

## III. Attacks and Defenses

Attacks from adversaries against Tor can be classified into several categories: attack on the user, attacks on their computer, attacks on the Tor network itself, attacks on the exit's connection across the Internet, and attacks on the server. Tor's developers have implemented or recommended various defenses against common attacks on each of these layers.

### *A. Weaknesses introduced by the user*

Users on Tor must take precautions when using Tor. Although Tor's onion routing provides privacy and anonymity to their traffic, users can identify themselves by revealing their real name or information about where they are. If they build an electronic trail that can be used to deanonymize them, the use of Tor is rendered useless by their carelessness.

(going to expand this a bit more)

### *B. Attacks on user's computer*

Tor users can also be revealed or have their privacy compromised by their own workstations. If their computer contains spyware, backdoors, or other malicious software that can be used to identify them or their traffic, then a user may be tricked into believing that they are anonymous when in fact they are not.

(going to expand this a bit more, discuss Javascript vs NoScript, cross-site scripting attacks, TBB defenses)

### C. Attacks on the Tor network

The relay chosen for the first hop in the circuit has knowledge of the user's IP address, can thus deanonymize them in that respect. Although this first relay does not know the rest of the circuit and cannot deduce the final traffic exiting Tor, the knowledge of the user's IP can be problematic if the relay is operated by an attacker. With this information, an adversary would be able to trace users or deny them entry into the Tor network.

Tor combats this through *entry guards*. The Tor client software chooses three relays out of the pool and then exclusively uses one of these chosen relays as the first hop of the circuit. This protects the end user against the ~~p~~predecessor attack~~,~~ wherein the attacker can preform end-to-end correlation and deanonymize the user if compromised nodes are chosen for the first and last hop in the circuit,[1] and they make it more difficult for an attacker to add some relays to the Tor network and immediately begin tracing users. As of the time of this writing, there are 4,737 relays on the Tor network, so the chances of a client including a compromised node in its selection pool is 4,737 choose three without replacement, or 0.06334%.

Tor is susceptible to timing attacks. To achieve its low-latency objective, Tor does not explicitly re-order or delay packets within the network.[6] Therefore, if an adversary controls both the first hop and the final hop in the Tor circuit then they can use statistical timing tests to become reasonably confident in identifying the activities of the end-user.

(going to expand this a bit more, add some citations)

### D. Compromising the exit node

Exit nodes are valuable on the Tor network due to their necessity for the final hop of the circuit and because they are rarer than traditional relays. Exit node operators typically have to deal with ISP complaints for any nefarious Internet activities performed by the users at the other end of the circuit. Although these complaints can be largely combated by implementing a reduced exit policy, there are 912 exit notes at the time of this writing, 19.74% of the total 4,737 relays on the Tor network. It is entirely possible than an adversay could control an exit node, giving them insight into the traffic leaving Tor. If the traffic is encrypted, the exit node could still see the DNS lookup and HTTP headers. If encryption is not implemented, all traffic would thus be available for inspection.

In September 2007, Swedish security consultant Dan Egerstad operated five exit nodes and monitored their traffic. Over the course of his experiment, he obtained the usernames and passwords for over 100 email accounts, read the correspondence from embassies in Australia, Japan, Iran, India, and Russia, along with communication between dignitaries including the Indian ambassador to China, various politicians in Hong Kong, workers in the Dalai Lama's liaison office, and several human-rights groups in Hong Kong. He also obtained sensitive spreadsheets and documents about military and political activities. Egerstad's analysis further revealed that approximately 95% of the traffic flowing over his nodes was not encrypted.[3]

An adversary could operate one or more exit nodes, as Egerstad did, or they could wiretap the exit's traffic and gain access to the traffic that way, perhaps without the knowledge of the exit operator. While the exit has no knowledge of the entire circuit and thus cannot deduce the origin of the traffic, the unencrypted traffic can be quite revealing and problematic, as Egerstad demonstrated. The primary way that Tor combats this through encryption. The Tor Browser Bundle comes with the HTTPS Everywhere, which prefers the use of HTTPS when connecting to web servers. Most email servers also support SSL/TLS, and users can also manually encrypt their documents or emails using PGP. Although Tor has no way of enforcing this final layer of encryption, it highly recommends it and places notices about this on many places on its website.

### E. Attacks on the web server

The web server can also be a point of failure for the privacy and anonymity of the end user.

(going to expand this a bit more. Does the web server support SSL/TLS? The adversary could fake masquerade as the server)

## IV. Adversaries

(going to expand this a bit more, discuss NSA, FBI, Silk Road, China, etc)

## V. Analysis

(going to expand this a bit more and devote the couple pages to this section. I've got IEEE citations like [8], [11], and [12] that should be very helpful in demonstrating some weaknesses of Tor

## VI. Conclusion

(obviously I need this. Summarize design, attacks, defenses, analysis, reviews in the literature, and then provide my opinion about the future)

## References


[1] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields *The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems*. ACM Transactions on Information and System Security (TISSEC) 4(7), November 2004, pages 489-522

[2] Abdelberi Chaabane, Pere Manils, Mohamed Ali Kaafar *Digging into Anonymous Trafc: a deep analysis of the Tor anonymizing network*. IEEE, 2010

[3] Kim Zetter *Rogue Nodes Turn Tor Anonymizer Into Eavesdropper's Paradise* Wired, 2007

[4] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid *Recommendation for Key Management, Part 1: General (Revised)* National Institute of Standards and Technology, March 2007

[5] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, Douglas Sicker *Low-resource routing attacks against tor*. ACM, 2007

[6] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, Douglas Sicker, *Shining Light in Dark Places: Understanding the Tor Network*. Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195-2969, 2008.

[7] Nikoli, Ivica *Distinguisher and Related-Key Attack on the Full AES-256* Advances in Cryptology, 2009

[8] Liu Xin, Wang Neng *Design Improvement for Tor Against Low-Cost Traffic Attack and Low-Resource Routing Attack* 2009 International Conference on Communications and Mobile Computing

[9] National Security Agency *The Case for Elliptic Curve Cryptography*. NSA, 2009
[10] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger *Biclique Cryptanalysis of the Full AES* Microsoft Research Redmond, 2011
[11] Zhen Ling *Equal-Sized Cells Mean Equal-Sized Packets in Tor?* IEE International Conference on Communications, 2011
[12] Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fuc, Weijia Jia, Wei Zhao *Protocol-level attacks against Tor* Computer Networks, 2012
[13] Robert Graham *Tor is still DHE 1024 (NSA crackable)*. Errata Security, 2013
[14] Overview of Tor
[15] Tor FAQ