Mark Cummins
ICS-451
4 December 2016
Ravi Narayan

Tor and the Quest for Anonymity

In the mid 1990's, with the web and its protocols well established, a question was begotten: "Can we protect our communications in this architecture?" This question was asked in the United States Naval Research Laboratory with the objective of protecting important online communications for the United States intelligence community. Thus, onion routing was born, the alpha version of Tor in 2002. Tor became publicly available to all users in 2004. From 2006 and beyond, the Tor project would be under a nonprofit for its future development. Tor has been sensationalized in the mainstream media from users such as Edward Snowden who revealed top secret government files to the infamy surrounding the darknet; bringing to light Tor's hidden services like the Silk Road marketplace and other illicit services. What isn't publicized is the majority of Tor users that are normal people whose only crime is practicing civil disobedience, circumventing oppression, and protecting their identity.

Before diving into the rabbit hole that is the Tor protocol, things must be understood before being caught up in the jargon and the rather futile nature of achieving pure and complete anonymity on the web. Tor doesn't make you completely anonymous, that wasn't its intention. Instead the primary design was to make it harder to trace an individual by clouding their steps. Therefore the correct term is anonymizing rather than anonymous. A perfect analogy to Tor's design for anonymity is military camouflage designed to make an operator invisible. Military camouflage might render an operator invisible in the sense that's he's undetectable to the naked eye. But with tools like a thermal scope that can track your heat signature, the effectiveness of camouflage is nullified and leaves the operator vulnerable. With modern tools, organizations and individuals can exploit Tor leaving the user traceable, completely nullifying Tor's effectiveness.

The objective of this paper is two things. The first is explaining Tor in such detail that someone who has never heard of Tor can fully wrap their mind around it. The second objective is to begin the discussion of achieving maximum anonymity. Tor does not make you completely anonymous. An analogy of Tor is moving to a city where nobody knows you. As you start to frequent restaurants, register things in your name, and use your credit card, you become known, and your anonymity goes away. This is what happens if you buy things on Tor, frequent sites where you have a user name or give signs to identify yourself. Also, if you log onto emails, social media, and other sites where you use your real name, your anonymity is gone.

The objective of Tor is to provide a circuit based, low latency communications service that emphasizes anonymity of the user. This is done by utilizing perfect forwarding secrecy, congestion control, integrity control, authentication, and a simple method of hosting your own server within the hidden services. It also aims to do these things while maintaining scalability and usability so that more users use Tor. This is because Tor users achieve greater anonymity as the user base and number of onion routers grow. Tor provides the same services that users are familiar on the clear web such as web browsing, instant messaging, email, and secure shell. Between your computer and the server you wish to access, your request is bounced through onion routers that make it difficult for the server to trace it back to the origin of the request. This bouncing of your request through onion routers is called a circuit. The circuit is created by your Tor browser making a request with onion routers to generate it. The circuit will always consist of three onion routers to relay your information from each other to layer your identity in anonymity.

The circuit achieves a high level of anonymity by using perfect forward secrecy. This means that each onion router doesn't have full knowledge of the circuit topology. It only knows its immediate source address and the immediate destination address. This is accomplished by the initiator of the circuit creating unique symmetric session keys between each onion router in the circuit. This is seen as the layers of encryption that anonymizes the user. What strengthens the circuit is the fact that a TLS link is generated between a router's receiving and sending end. Whatever type of application request that's sent to a router will be handled accordingly, then sent to the next immediate router to perform the same task.

Each onion router holds onto its own two keys. An identity key and an onion key. The identity key is used for authentication to circumvent man in the middle attacks. The identity key will sign TLS certificates and the summary of an onion router's router descriptor. The router descriptor consists of a summary of the router's keys, address, bandwidth, and exit policy. The onion key is the key that decrypts that onion's layer of encryption in the circuit. The onion key is also used to negotiate or instantiate ephemeral keys. In addition to these keys, the TLS protocol will generate its own link that lets onion routers communicate between each other.

The Tor packet is layered over the TCP stream. The Tor packet are a fixed size of 512 bytes in length and are known as cells. The header of the cell consists of a circuit identifier that tells which circuit that cell belongs to. The header also contains a command that tells the cell what to do with its payload. The commands designate whether or not the cell is a control cell or a relay cell. Control cells are interpreted by the onion router that receives them. They could tell the router create, destroy, or pad a circuit. Relay cells have an additional header called the relay header that simply carries stream data to another router. The relay header consists of a stream ID, a checksum, and the actual relay command. The stream ID determines which stream to go through since it's possible for many streams to go through the same circuit. The checksum verifies the command has maintained integrity as it's relayed from one end to the other end of the circuit. The relay header is encrypted with the 128 bit AES for each hop. As it goes through the circuit, each layer is decrypted. The relay commands could tell the router to relay data, begin, end, teardown, connect, extend, truncate, sendme, or drop. Relay data informs the router that data is being pass through the stream. Relay begin opens a stream, end closes a stream, and teardown closes a broken stream in the event there is one. Relay connect tells the initiator that a relay start was successful. Relay extend adds another hop to the circuit, truncate tears down a hop on the circuit, sendme is used to relieve network congestion, and drop is used to implement dummy relays. An example of a control cell would be a two byte circuit identifier, a byte sized command, and a 509 byte payload. An example of a relay cell would be a two byte circuit identifier, a byte sized relay, a two byte stream ID, six bytes that control congestion, two bytes that show the length of the cell, a byte that has the actual relay command, and 498 bytes dedicated to the payload.

For speed, a user's Tor browser premakes circuits before a request is made. Because of the short amount of time required to create new circuits, your Tor browser decides whether or not to jump on a new circuit every minute. Thus, if your current circuit failed, you would be onto another one without noticing a wane in performance.

The circuit is the foundation of the Tor network. A user's Tor browser incrementally creates its circuit in tenths of a second. The Tor browser negotiates a symmetric key with the onion router, then hops to the next one to do the same thing until a circuit is created. Let's designate the Tor browser as Mark, the first, second, and third onion router as alpha, beta, and delta respectively. To create the first part of the circuit, Mark sends a create cell to alpha. The create cell will have a unused circuit identifier and the first half of the Diffie-Hellman handshake in the payload. This is encrypted with alpha's onion key. Alpha responds to Mark with a created cell response signaling that the circuit between them is made as well as the other half of Diffie-Hellman handshake shared with Mark and a hash of the negotiated key. With the negotiated key created, the circuit between Mark and alpha is created, and relay calls can now be encrypted between them.

To continue building the circuit, Mark will send a relay call to alpha to extend the circuit, specifying the address of onion router beta with half the Diffie-Hellman handshake for beta. Alpha will put the Diffie-Hellman handshake half into a create command and send it to beta. Alpha will also make a circuit identifier for the circuit built between him and beta that Mark doesn't know about. Beta will reply back with a created cell, alpha will put the reply over his relay extended cell and reply back to Mark. Now Mark and beta have a shared key between them that alpha doesn't recognize.

Now Mark will continue building the circuit once more. Mark will send a extend relay call that's under a relay data command to send to alpha. Alpha will decrypt the first layer and send the extend relay call to beta. Beta will then put the Diffie-Hellman handshake half into a create command and send it to delta. A circuit identifier will be made between beta and delta that Mark and alpha have no part in. Delta will reply with a created cell to beta and an extended cell will be forwarded through alpha to Mark. Mark and delta will now have their own shared key. Mark now has a complete circuit for him to browse with.

Because of the relay process, beta and delta don't know that Mark made the call to create a circuit with them in it even though they both share their own private key with Mark, thus achieving unilateral entity authentication. Also, since Mark has a different key with each onion router, he's able to authenticate each onion router independently from the other onion routers.

With the circuit established, Mark can set up a relay cell. He'll do this by creating the header then layering it with a relay data cell command for each onion router to follow along with the layers of encryption in the proper order. With this circuit, alpha will receive a cell from Mark. Alpha will decrypt his layer of encryption with his shared key to find the relay call for him to follow. It'll tell him to relay it to beta, so he'll look up beta's circuit ID and relay it to him. Delta will check that his decrypted header has a valid digest to verify integrity from the original sender.

Two essential designs implementations in Tor are integrity checking and congestion control. Integrity checking ensures that an attacker can't decrypt cells and change the commands in them. This also ensures that cells that hop from one end of the circuit to the other end maintain integrity. The difficulty in this is that each onion router in the circuit can't do the integrity check since they don't how to decrypt other layers of a cell except their own. Thus, the check is only done at the edges of a stream. Just like on the normal web, congestion control is very important so that more time can be used sending packets successfully rather than dropping packets and re-sending them. Since it's possible for multiple users to use the same onion routers in their circuit, those onion routers can get congested. Because of how the Tor network is set up, this can affect the speed of the entire network and bottleneck certain areas of it. Since Tor doesn't use the TCP header and utilize its congestion control window, it makes the onion routers keep track of the packaging and delivery windows. The packaging window determines how many relay cells the onion router can package from an incoming TCP stream back to the Tor user. The delivery window is how many relay cells the onion router is willing transmit out to the network.

What complements the Tor circuits are the rendezvous points. These allow users to set up servers within the Tor network, making them accessible only through the Tor browser. Like Tor users, rendezvous points allow servers to have their IP addresses hidden as well. These location hidden services follow four techniques for them to stay up, be authenticated, and remain anonymous. The first one is access control, this gives the owner of the server the ability to filter requests to their hidden service. The second one is robustness, meaning that the hidden service is not tied to a single onion router. Numerous onion routers should be able to host his server so that the failure of one doesn't take down the service. The third is smear-resistance, even though the origin of the hidden service is kept anonymous, someone else shouldn't be able to imitate you or lead users of your hidden service to a fake rendition of your service. The final point is application-transparency, meaning users shouldn't have to modify their applications like forcing the user to use HTTP instead of HTTPS, rendering the connection insecure. Unlike Tor users that have one entry way, Tor hidden services advertise their service to numerous entry onion routers, allowing the hidden service to be easily accessed.

For Mark to access sigma's hidden service, Mark will first designate some onion router as his rendezvous point. Mark will then connect to one of the onion routers hosting sigma's hidden services and inform sigma to meet him at the onion router designated as the rendezvous point. This is done so that the onion routers that host the hidden service don't also carry the responsibility of providing the service as well. A typical scenario will follow like this: Sigma will generate a public key for his hidden service so users can securely access it. Sigma will then choose onion routers that'll host or advertise his service, authenticating this process by using his public key as a signature. Sigma will then build circuits for each onion router that advertises his website so they can accept user requests. From here Mark will hear about the hidden service and build a circuit to a rendezvous point. Mark will then anonymously stream to one of the onion routers advertising sigma's hidden service with a message encrypted with sigma's public key along with the location of the rendezvous point, rendezvous cookie that'll authenticate sigma, and the first half of the Diffie-Hellman handshake. Decrypting the message and reading the request, sigma will then build a circuit to the same rendezvous point and sends the rendezvous cookie, the second half of the Diffie-Hellman handshake, and a hash of the shared key to Mark. This will let Mark know that he shares a key with sigma. Anonymity is maintained in this interaction by the rendezvous point connecting Mark's circuit to sigma, not knowing who Mark or sigma are or what they're communicating about.

The quest for true anonymity on the Internet is an everlasting pursuit. From social media to public and private organizations secretly collecting your metadata, the demand for anonymity is justified. In this context, the case for anonymity is applied to those that benefit most from it and rely heavily on it. Users that are whistle blowers, human rights activist, and critics of government have an ethical reason to want to be anonymous. Without it, they are defenseless against those that wish harm on them. Even if the user is not someone who is in danger without it, they should be aware of how their privacy is being violated online. To put it in layman's terms, just because you live a very safe neighborhood doesn't mean you should keep your front door unlocked. Your house will never be invaded, until it happens. Thus, with the whistleblower with active pursuers on him in mind, the quest for true anonymity will start.

Anonymity does not end with Tor, it is one brick in the wall. With tools like Quantum Insert and software like the Great Firewall of China, the user needs to take many steps, often inconvenient, to circumvent them. The law of diminishing returns applies to obtaining anonymity. For example, to achieve anonymity in the 90th percentile will require an ounce of effort. The next ounce of effort will yield a five percent increase in anonymity. The next ounce gets you the 98th percentile, the next to 99, and the subsequent ounces will yield decimals of a percent approaching 100 percent. By incorporating other things that increase anonymity or simply prevent your anonymity from being revealed, Tor increases its resistance to exploitation and your anonymity percentile increases.. The first method of circumvention is not technical, not utilizing anything that can be identified with you. This includes credit cards, using your social security number, personal emails, social media, and other types of online accounts. You could use every tool to anonymize yourself and completely nullify them by using your credit card to purchase something. Alternatives like bitcoin and bitcoin laundering allow a user to make transactions while keeping their identity hidden.

The most important aspect in conjunction with Tor to ensure your anonymity is with your operating system. The amnesiac incognito live system Tails was built with using Tor in mind. This makes it so that your operating system doesn't become the weakest link. Cases like the flash player in your operating system revealing your IP address even when using Tor are eliminated as possible exploitations. The use of Tails brings your anonymity to the 90th percentile. The journey to the 95th percentile of anonymity is with virtual private networks which has two cases depending on the goal of the user. If you used Tails to sign into a VPN to log into the Tor network, this would allow you combat the Great Firewall of China if your VPN is esoteric. Because Tor is difficult to access in China on its own, this setup would allow you to circumvent the censorship. The other case is using Tails, logging into Tor, then signing into a VPN from the Tor network, allowing you to access websites that block Tor users and to hide your geological location. As long as your VPN doesn't keep logs, both cases are useful in preventing your identity from de-anonymization tactics. An obfs4proxy is another useful tool as it makes your Tor traffic appear as normal network traffic. Network traffic monitors are able to distinguish Tor traffic because of its unique packet compared to other protocol packets.

Subsequent increases in anonymity now carry the burden of inconvenience. This is because the user won't be able to use Tor within the comfort of his home. From here, if the user is very concerned with their anonymity, extreme steps are taken. One of them would be mixing up your real IP address. This is done by rotating around places that provide public internet and even going so far as to rotating around locations that have geologically different IP addresses. Another layer of anonymity would be to boot Tails from a different computer every time. The most extreme steps taken to maximize your anonymity would be to not exist. This could go as far as faking your death or stealing someone else's identity alive or deceased to throw the scent off of you. These steps may or not be necessary depending on your goals and the level of anonymity needed to reach that goal. In the case of Edward Snowden, most if not all of these steps were taken when he leaked the information on the NSA. He even went as far as to include world renowned journalists and Wikileaks to legitimize his whistle blowing.

Anonymity defeatists argue that it's futile to achieve true anonymity on the web. Since the web's architecture wasn't built with anonymity in mind, you would have to build an entirely new web with anonymity close in mind. At the moment, the defeatists have a point, but the web wasn't built with the scalability and foresight of having billions of machines using it during its initial development, yet it's doing it. Thus, as the Tor project improves and the demand for anonymity on the web becomes more mainstream, true anonymity can be a reality without the use of extreme measures.

So how does Tor match up against contemporary tools like Quantum Insert and the Great Firewall of China? The first conducts surveillance on you and the other censors you from certain websites in China. The first contender, Quantum Insert can win or lose against Tor depending on the circumstances. If the NSA can track you through the Tor network from end to end correlation and own the onion routers that you create a circuit with, then they can perform the Quantum Insert raceway condition. But, since it's unknown how the malware will perform in the Tails operating system, your IP address may or may not be traced. It's also unknown whether or not the Quantum Insert malware will poison your bootable USB since Tails is never installed natively on a computer. If you ran packet capturing software like Wireshark to see if you had an HTTP GET response duplicate, you might be able to detect a Quantum Insert attack being deployed on your computer, allowing you to abandon your computer if you suspect an attack occurred.

The Great Firewall of China fares the same way against Tor. By itself, Tor is no match against the great firewall because it's configured to block Tor packets. To achieve some success, you would have to boot up Tails, (assuming you could even download it or access the Tails website from China) access a VPN that's not blocked by the firewall, get on the Tor browser, and run an obsf4proxy. Because the great firewall is very sophisticated, it's able to distinguish onion route relays and block them. What the user needs to do is make sure your circuit connects to Tor bridges that aren't publicly listed. This is not a permanent solution, as the great firewall can update to deny your esoteric VPN and be able to distinguish Tor traffic from regular traffic. The solution to this is an obfs4proxy, a pluggable transport that makes your tor traffic look like normal traffic to someone monitoring you. It's also not certain that even if you follow this routine, you will be able to access clear net websites like Facebook and Google that are blocked in China. This routine is moreso geared towards accessing Tor hidden services. The firewall cannot block specific onion sites as their IP addresses are hidden (they also use circuits).

Tor and the quest for anonymity will run in parallel with each other as both have a common goal. The design of Tor is robust and strict but not perfect. The result of a foundation that wasn't built with security in mind makes remaining anonymous a difficult task. Anonymity can progress in two concurrent ways: the Tor protocol being improved over time and a divergence from traditional network architectures to peer-to-peer networks like the content-addressable network. Decentralization and the minimization of control from a single entity (internet service providers) will strengthen anonymity. Tor is still in its infancy, and as more tools are used in combination with it and more research is conducted in network anonymity, the 100th percentile can become a reality.

Citations:

https://www.bestvpn.com/blog/42672/using-vpn-tor-together/

https://tails.boum.org/blueprint/vpn_support/

https://en.wikipedia.org/wiki/Tor_(anonymity_network)

http://blog.zorinaq.com/my-experience-with-the-great-firewall-of-china/

https://www.technologyreview.com/s/427413/how-china-blocks-the-tor-anonymity-network/

https://blog.cloudflare.com/the-trouble-with-tor/

http://conferences2.sigcomm.org/imc/2015/papers/p445.pdf

http://motherboard.vice.com/read/what-firewall-chinas-fledgling-deep-web-community

https://zqktlwi4fecvo6ri.onion.to/wiki/index.php/Tor

https://www.eff.org/torchallenge/what-is-tor.html

https://www.eff.org/torchallenge/faq.html

http://freenuts.com/3-ways-to-get-bridges-for-tor/

http://www.bsdnow.tv/tutorials/tor

https://svn.torproject.org/svn/projects/design-paper/tor-design.html#sec:rendezvous

https://www.torproject.org/docs/faq.html.en#Proxychains

https://www.torproject.org/docs/pluggable-transports.html.en