Alice in Onion Land:

On Information Security of Tor

Ville-Valtteri Immonen

Master's thesis

School of Computing

Computer Science

June 2016

ITÄ-SUOMEN YLIOPISTO, Luonnontieteiden ja metsätieteiden tiedekunta, Joensuu
Tietojenkäsittelytieteen laitos
Tietojenkäsittelytiede

Opiskelija, Ville-Valtteri Immonen: Sipulireititys ja tietoturva
Pro gradu -tutkielma, 41 s.
Pro gradu –tutkielman ohjaajat: FT Markku Tukiainen
Kesäkuu 2016

Kryptoverkkojen tarkoitus on luoda verkkoon kommunikaatiopolkuja jotka mahdollistavat viestien lähettämisen tavalla jolla viestien alkuperää ei pystytä yhdistämään niiden lähettäjään. Tor –sovellus käyttää sipulireititys –nimistä tekniikkaa viestien salaamiseen. Kyseinen tekniikka saa nimensä tietorakenteesta, jonka sisällä informaatio kulkee lähettäjältä vastaanottajalle. Yhteyden alulle paneva kone muodostaa sipulireitityspolun ja paketoi lähetettävän viestin sipulin sisään ennen sen lähettämistä eteenpäin polkua pitkin. Sipuli muodostetaan kryptografisesti salatusta kerroksista, jotka jokainen kommunikaatiopolun varrella oleva reititin kuorii yksitellen.
Tor pyrkii pysymään matalan latenssin palveluna ja tekniikka on suunniteltu tämä periaate mielessäpitäen. Erilaiset puolustuskeinot, kuten aloitusvartijat ja sillat tarjoavat tietoturvaa samalla kuitenkin pitäen verkon laskentatehon matalana. Ensimmäinen versio sipulireitityksestä kehitettiin vuonna 1996 ja viimeisin vieläkin kehitteillä oleva, kolmas sukupolvi, julkaistiin vuonna 2006.
Tässä tutkielmassa tarkastelemme sipulireititystä: miten eri sukupolvet eroavat toisistaan, millaisia tietoturvauhkia sipulireititykseen liittyy ja miten niitä vastaan voidaan puolustautua. Tarkastelemme kahdeksaa viime vuosina julkaistua artikkelia tietoturvahyökkäyksistä, eli teoreettisista tavoista, joilla sipulireititysteknologian tietoturva voidaan murtaa. Tämän jälkeen jaamme nämä hyökkäykset kategorioihin. Ehdottamamme kategoriat tietoturvahyökkäyksille ovat: Lähtö- ja loppusolmujen valintaa hyödyntävät hyökkäykset , Liikenteen ja ajoituksen analyysiin perustuvat hyökkäykset, Autonomisten järjestelmien tason ja globaalin tason hyökkäykset ja Sovellustason hyökkäykset.

Avainsanat: Sipulireititys, Tor, Anonymiteetti, Tietoturva, Tietosuoja

ACM-luokat (ACM Computing Classification System, 1998 version): C.2.1

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Joensuu
School of Computing
Computer Science

Ville-Valtteri Immonen: Alice in Onion Land: On Information Security of Tor
Master's Thesis, 41 p.
Supervisors of the Master's Thesis: PhD Markku Tukiainen
June 2016

Abstract: Cryptographic networks aim to create communication paths through internet that allow sending messages in a way that the origins of the message cannot be linked to its sender. Tor -application uses a technique called onion routing to encrypt messages. The technique gets its name from the data structure inside which information travels from sender to receiver. The machine that initiates the connection forms an onion routing path and wraps the message inside an onion before sending it down the path. An onion is formed from cryptographically encrypted layers that are peeled one by one by each router along the communication path.
Tor aims to stay a low latency anonymity service and the technology has been designed with this principle in mind. Different defences like entry guards and bridges offer protection while keeping the computing cost of the network low. First version of onion routing was developed in 1996 and the last, third generation that is still under development was published in 2006.
In this thesis we examine onion routing: how do different generations differ from each other and what kinds of data security threats onion routing has and what possibilities is there to defend against them. We discuss eight papers published in the recent years about theoretical attacks, meaning theoretical ways to break the security provided by onion routing technology and divide these attacks into categories. The categories we propose are: Entry and exit onion router selection attacks, Traffic and time analysis based attacks, Autonomous system level and global level attacks, and Software-level attacks.

Keywords: Onion Routing, Tor, Anonymity, Information Security, Information Privacy

CR Categories (ACM Computing Classification System, 1998 version): C.2.1

Abbreviations

AS	Autonomous system. A part of a network that is controlled by a single administrative entity.
CircID	Circuit identifier of a cell that tells to the node which circuit the cell belongs to.
CMD	Control cell command. Reports the type of a cell so that the receiving node can interpret it correctly.
DHT	Distributed hash table. Used by applications like BitTorrent to store information about nodes in the P2P network.
DoS	Denial-of-service attack. An attack that tries to stop a server from responding to clients by flooding it with messages.
EFF	Electronic Frontier Foundation. Organization that funds the development of onion routing today.
HTML	Hypertext Markup Language. The markup language used to present webpages.
HTTP	Hypertext Transfer Protocol. Application level protocol used to transfer webpages.
HTTPS	A protocol that uses HTTP over SSL to create a trusted connection between a server and a client.
img	HTTP image tag. Tells a browser to retrieve and insert an image into a webdocument.
IP	Internet Protocol. A protocol designed to connect computers in the internet with unique addresses.
NRL	United States Naval Research Laboratory. Developed the first generations of onion routing.
NSA	National Security Agency. Intelligence organization in US that has tried to crack the Tor encryption.
ONR	Office of Naval Research. Funded the development of onion routing from 1995 to 2004.
P2P	Peer-to-peer. A distributed application consisting of multiple peers over a network.
PEX	Peer exchange. A protocol used by BitTorrent to control file saving between peers.
TCP	Transmission Control Protocol. Transport layer protocol to deliver streams over an IP network.
Tor	Software designed to allow the forming an anonymous connection inside the internet.
UDP	User Datagram Protocol. Transport layer protocol to deliver streams over an IP network.

Table of contents

1 Introduction

Since the invention of sending written down messages between people there has also been a need to prevent other people from capturing and reading those messages. When alongside physical messages came electric communication and the internet, the problem certainly did not go away. Huge amounts of effort is being made to develop efficient ways to scramble, or encrypt, messages so that only the communicating parties know what the contents of the messages are. Normally the messages traveling through the internet are encrypted, and only their sender and receiver are known to everyone, but in some situations even this information is wished to be kept as a secret.

The term *traffic analysis* is used to describe a process that tries to study messages going through an information network with the attemption to make conclusions about the traffic based on the collected knowledge [36]. Every action made in the internet sends a series of messages through multitude of servers that can be located anywhere in the world, making the actions quite public. This means that is relatively easy for someone to monitor the ongoing traffic. Eavesdropping on the traffic is not hard. As the data goes through the public internet, it travels through several locations and someone trying to eavesdrop on this communication only needs to be in one spot along this communication path to observe this traffic.

An IP packet that transfers data through the internet has two parts: firstly, the data being transmitted, which is called *payload*, and secondly, a *header* that contains metadata about the payload. The payload is usually encrypted before it is sent forward, but the header part is not, and it tells a lot about the data, such as the sender, the receiver, the time and the size of the data block. This creates an information security problem for individuals and organizations, because in general anyone intercepting these messages could observe the traffic and form models about the traffic based only on the headers of the network packets.

Because of the reason explained above, an increasing part of internet users do not want for outsiders to be able to look what they are doing in the internet and this has created a demand for applications that could offer anonymity. Users and their reasons

for this vary: ordinary people want to protect their privacy from marketers and irresponsible companies, journalists and their audience want to guarantee freedom of speech without political censorship, law enforcement officers want to study internet crimes discreetly and army field agents want to protect themselves against enemy espionage during operations [39].

Various cryptographic networks try to prevent traffic analysis by mixing the messages traveling through a network in such a way that an external observer cannot trace the origin and destination of a message. The aim is to allow a completely private way of communicating over the internet. One of these anonymity-offering crypto networks is *Tor* [8] which uses a technology called *onion routing* to encrypt messages. The technology, which was originally designed for the purposes of the United States' army, is in the present day being developed by enthusiastic programmers and scientists. Nowadays onion routing is actively used to avoid internet censorship, to prevent eavesdropping and to assist freedom of speech.

This thesis takes a closer look at onion routing: what kind of measures does it take to protect messages against traffic analysis and also what kind of countermeasures can be taken to break this protection. We take a look at some recent research papers describing different types of attacks against the system and finally propose some categories which can be used to classify different types of attack methods. With these categories we are able to create an attack model that an adversary could follow.

The thesis is organized as follows: second chapter looks at how in general onion routing works. Third chapter goes through the history of the technology to explain how the design of the technology has taken shape over time. Chapter four lists different kind of attacks against onion routing and provides a classification of these attacks. Final chapter five presents a conclusion with some final thoughts.

2 Onion routing

The history of onion routing begins from the year 1995, when the Office of Naval Research, or ONR, in the United States started to finance a project that was aimed to remove identification from routing. In other words their aim was to create a way of creating connections in the internet anonymously [28]. At the beginning three persons, David Goldschlad, Michael Reed and Paul Syverson began the development of technology that was later going to be called onion routing. The initial result of the project was the first prototype in the year 1996 and after this the technology has been developed for another two generations.

Before Goldschlad, Reed and Syverson started developing the first prototypes, they had a picture about what kind of principles the technology would follow. Firstly, the code should be open to everyone. For the users to be able to trust in the system, they would have to see the code they are running. The second main principle of onion routing is to allow free riding. This means that the computer that is used to access the system does not necessarily have to be made a server. This principle has some security advantages, which will be analysed later.

2.1 Overview of the technology

Onion routing gets its name from the data structure inside which the information travels through the network [8]. An *onion* is built from cryptographically encrypted layers by the sender of the message. Every router along the communication path peels these layers one by one, as the message travels through the network. Incoming messages travel along the same path, except this time the routers are adding layers to the onion, which the receiving end then peels.

As noted earlier, onion routing is based on creating a cryptographic route, through several routers, between two computers. This path is called an *onion circuit* or *onion chain* [28]. The routers that form the path are selected randomly from all servers registered inside the onion network and the circuit is formed in a way that every node knows only the next and previous router in the communication path. The initiator of the path makes the decision about which routers will be included in the chain.

To prevent a situation where all routers participating in the communication path would be able to open the messages traveling along the chain, the initiator must distribute unique, symmetric encryption keys individually to every involved router. The reason for using symmetric keys is that symmetric encryption uses the same key for encrypting and decrypting messages so messages can be sent to both directions with the same key. Symmetric encryption is also computationally much cheaper than public-key encryption.

The symmetric encryption keys are distributed by using public-key encryption. Public-key encryption uses two keys: a private key (which is not known by others) and a public key (can be known by everyone). The messages that are sent to the receiver are encrypted with a public key, which in turn cannot be used to decrypt the messages. The receiver can open the message with their private key. Because every router has their own public key, the symmetric keys that are used for creating the onion can be distributed safely by using public keys without the danger of other routers finding them out.

The reason for using symmetric encryption for creating the actual onion is that symmetric encryption demands a lot less computing power than public-key encryption. There are loads of messages traveling through the network in all directions and every message has to be processed by an encryption algorithm. By using symmetric encryption instead of public-key encryption, the computing cost of the network can be reduced significantly.

The algorithm for forming an onion chain is described in detail in the third chapter. Next we will take a closer look at the evolution of the onion routing technology. As mentioned earlier, onion routing has been developed incrementally in generations. The features mentioned so far are not true for all three generations, but were included in the technology one by one. The differences between the evolving generations will be the main subject of the second chapter.

2.2 The evolution of onion routing

When the designing of onion routing was set to motion in the U.S. Naval Research Laboratory, or NRL, Goldschlad, Reed and Syverson had just a few basic principles,

and some guesses about in which kind of situations the technology would be used, to guide them in their work [28]. It was not until the first generation was published when they got their first illustration of how functional the design actually was. Features were added and refined in the following generations according to the observations made by the developers and also according to feedback coming from the users. The generations therefore represent the different development phases of onion routing and by examining the generations it is possible to see why the onion technology is implemented the way it is today.

Syverson, who was one of the first people developing onion routing, gives a nice picture about the ideas and philosophy of onion routing in his article *A Peel of Onion* (2011) [28]. The article is a must read of everyone interested about onion routing and it has been one of the main sources for this Master's Thesis in addition to the article *Tor: The Second-Generation Onion Router* (2004) [8] by Dingledine, Mathewson and Syverson. This chapter has been mainly written based on these two papers, and gives only a summary of the main points of the story.

2.2.1 First generation (1996 - 2004)

The basic idea of onion routing is to allow making an anonymous connection from within the onion network to servers which are external of the onion network. In addition to this it was decided to make possible to form a connections between two routers connected to the onion network. In both of these situations the length of the chain, meaning the number of routers in the chain, the sender and receiver included, has a big relevance from the aspect of information security.

In order for one of the nodes in the chain to break the anonymity, it needs to connect the anonymous messages to their sender and to their receiver. Basically a theoretical attacker needs to get the IP address of the sender and the receiver. Normally this could be done by looking at the headers of the data packages, but onion routing encrypts these headers. The most basic attack against onion routing is to set a router as an onion node and try to observe the messages that go through that router.

If the chain is three nodes long (sender, onion router and receiver), in case the middle node is the malicious attacker that wants to figure out the IP addresses of the two

ends of the chain, it will know these addresses immediately. When using four nodes (sender, two routers, receiver), the adversary needs to break the encryption of only one other node to break the anonymity of the chain. The developers did not see this as safe enough, so they decided to set the minimum and default length of the chain as five nodes.

So called *rendezvous servers* are used in the case that two machines that are both connected to the onion network want to send each other messages. The machines cannot form the connection path based on each other's IP addresses so the rendezvous server is used as a middle man to distribute the messages. Because both of the parties have to form their own onion chains between the rendezvous server, the length of the path with all routers included will be nine nodes. This feature has stayed the same for the next generations.

The largest difference between the first generation and the next two is that in the first generation version the client software and an onion router were totally integrated. As a computer joined the network to send messages, it also could be added as a part of an onion chain. This kind of functionality was against one of the main principles of onion routing and was changed as the second generation was released, making it possible to run a client and an onion router separately.

2.2.2 Second generation (2004 - 2006)

Various techniques were added to the second generation in an effort to increase information security. One technique is to send messages with random data, called paddings, along with normal traffic to make it harder for an outsider to analyse traffic. Second technique is to limit bandwidth to a constant rate, which eliminates identifiable changes in traffic flows. *Paddings* and *bandwidth-limiting* were thought to render traffic analysis harder, but these techniques were noted to be too computationally expensive in relation to the security they offer. Paddings and bandwidth-limiting were removed prior to third generation.

Some anonymity-offering crypto networks are based on mixing of components rather than forming an unpredicted route. These networks are generally called *Mix networks*

(developed by David Chaum in 1981) [8]. The purpose of mixing is to make it harder to connect incoming messages to crypted outgoing messages to each other.

Mix-technology was experimentally included to the second generation of onion routing. The purpose of this experimenting was to explore the security advantages mixing would offer. It was envisioned that adopting an already-widely-used method would make onion routing more acceptable to the wider audience.

Mixing was eventually abandoned in the third generation for the same reasons as paddings and bandwidth-limiting; it did not increase the information security enough and it increased the computing cost considerably. Doing the actual mixing is not computationally expensive, but adding mixing creates a risk of someone being able to eavesdrop to the messages and so far an inexpensive way of preventing this has not been found.

Another major change, in addition to mixing, was allowing the creation of circuits longer than five nodes. In this version the length of the circuits varied and in one chain there could be up to eleven nodes. By *tunnelling* the chains, or combining multiple chains, the path length could grow even larger. As with mixing and padding schemes, also this change was stated to be unnecessary and the number of nodes was restored to five in the next generation.

2.2.3 Third generation (2006 -)

Third generation is the last version of onion routing. The development onion routing is still on going, so the features may in the future change from what is stated here. We will however, look at the features of the third generation in the form as it was published in 2006. The differences between this generation and the previous one are smaller than the differences between first and second generations.

The biggest alteration from previous versions is in the protocol with which encryption keys are distributed to nodes when a circuit is first created. The first version used the onion -data structure for building the path [28]. This version uses *Diffie-Hellman protocol* to distribute the keys to the nodes incrementally. This implements so called *forward secrecy*, which means that even if an attacker could decrypt and

take possession of a single encryption key, they could not use it to open the other onion layers.

Another change in this version was the addition of directory servers. Before this change the information about network addresses was issued externally from the network. When the size of the network grows, this kind of way of giving out information becomes impossible and directory servers offer a flexible way of solving this problem. This also increases security, because the distributed information can be surveyed.

Next chapter takes a closer look at the actual implementation of the third generation onion technology. For recapitulation, table 1 shows a small technical comparison between the generations.

	Year published	**Mixing added**	**Length of chain**	**Client and server**	**Key distribution protocol**
1.Generation	1996 [28]	No	5	Integrated	Using onions
2.Generation	2004 [22]	Yes	1 --11	Separate	Using onions
3.Generation	2006 [22]	No	5	Separate	Diffie-Hellman protocol

Table1. The most distinctive features between different generations

3 Tor

First version of onion routing was published in 1996 [28]. When Office of Naval Research funded the project, they also maintained a website in which information of the progress of the project was published. This site contained technical information about the different generations and also scientific articles regarding onion routing. As ONR ended the funding in 2004, also the maintaining of the website was stopped and the site was shut down [22]. There is, however a copy of the site held in the address www.onion-router.net [22].

When ONR discontinued the financial support, Electronic Frontier Foundation, or shortly EFF, started financing the project. Paul Syverson and the onion team had time to release three onion routing generations under ONR funding. When the funder changed, a shift to open source was made and along that the new homepage has moved to the address www.torproject.org [35]. Both the old and the new website have been an important source of information for this thesis.

With the new website also an application called *Tor* was published. The purpose of Tor is to create a network that protects it's user from identification by using the onion routing technology. In other words, Tor is an implementation of onion routing in practice. From here on forwards, when discussing about onion routing, we are talking about Tor, the third generation onion routing implementation, and vice versa.

3.1 Tor Browser Bundle

The Tor application is written in C programming language, and its source code is maintained in a public repository in the address gitweb.torproject.org/tor.git. In order to use the application any user can download a package called Tor Browser Bundle, or TBB, from the projects website ([35]). TBB contains a program that allows the user to connect to the Tor network as a client and a special Firefox browser that can be used to connect to websites.

Tor Browser Bundle contains the necessary software for running a client and a relay.

TBB is preconfigured to act as a client and the user does not necessarily need to do any additional configuration to use the software [35]. There are however, some actions that the user can do to increase privacy such as turning on an extension called NoScript that turns JavaSript off from pages and using only HTTPS to form connections. To add a new Tor relay to the network the user needs to edit a configuration file with port info and exit policies.

Clients use the Tor software to form connections through relays. All the computers connecting to each other with TBB form a network, which is referred to as the *Tor network*. To work efficiently, the Tor network needs also other kinds of components despite clients and relays. Let's take a look at what other components Tor network has.

3.2 Tor network

In its simplest abstraction, an implementation of the onion routing network consists of clients that use the network to send messages, and routers, or relays, that receive onions, peel them and pass them along in the circuit to the next node. In addition to clients and relays, Tor network has for example name servers that keep track of servers that are registered to the network, and *hidden services* that can be used for example web publishing. Hidden services can be connected to only through specific Tor servers. [8]

Clients and routers form the bulk of the Tor-network [30]. Clients can also act as a router inside the network, but most of the clients are using the network just to form connections to other clients, hidden services and external services through other routers. The amount of clients using the network in March 2016 is around 2 million and the amount of relays in the network in 2016 is 7000.

To make Tor resistant to eavesdropping, along with normal onion routers, also so called *entry guards* have been added to the network [3]. The purpose of entry guards is to prevent attacks where a malicious party tries to place their router as the first node of a circuit. In the original implementation the first node was randomly selected from all routers that have been connected to the network which meant that as long as the connection were formed again and again for enough times, at some point the ma-

licious router would have been selected as the first node. Since the 2006 version three guard nodes instead of one entry node are chosen from a small number of reliable routers. These three guard nodes will be used for 30 to 60 days for all the circuits that the client forms, before discarding choosing new entry nodes [17]. This increases the network's ability to withstand eavesdropping attacks considerably.

Another measure to increase security is the addition of so called *bridges* [28]. For the purpose of increasing security some routers do not announce their addresses publicly. Bridges exist because normal, publicly listed Tor IPs get blocked by some firewalls. An individual who has no access to Tor network because of such firewalls can use bridges to go around these filters.

Also hidden services were mentioned earlier. These do not increase the information security as such, but the existence of hidden services is something that the developers wanted to enable. These services can be used for example to send instant messages between clients. When a hidden service wants to publicize its address, it forms some onion circuits and reports the last router's address of the circuit to name services. A client can form a connection to the service through *rendezvous nodes*.

3.3 Creation of an onion circuit

Let's take a closer look at how a circuit is formed. When a Tor user wants to form a connection to a server outside of the Tor network, their client software first downloads a list of all the registered Tor routers [3]. From these routers first an *entry node* is picked. Entry node is the first router in the onion circuit and it is chosen from the list of routers that the name service has announced to be fast and reliable. In case the chosen entry node is unreachable, a new entry node is selected to take its place.

After an entry node has been selected, the rest of the circuit is formed. The nodes that are listed to have the largest bandwidth and uptime are more likely to be added to the circuit. For the chain to be as stable as possible, only nodes that are diagnosed as stable are selected. Figure 1 shows a pseudo algorithm, copied from the paper *Low-Resource Routing Attacks Against Tor* by Bauer et al. [3], for choosing the exit router and the middlemost router in the chain. The algorithm differs from the current im-

plementation in such a way that the router cannot itself advertise the bandwidth, but the bandwidth is measured inside the network [26].

Algorithm: Non-Entry Router Selection

Input: A list of all known Tor routers,
router_list
Output: A pseudo-randomly chosen router,
weighted toward the router advertising
the highest bandwidth

$B \leftarrow 0$, $T \leftarrow 0$, $C \leftarrow 0$, $i \leftarrow 0$, *router_bw* $\leftarrow 0$,
bw_list $\leftarrow \emptyset$
foreach *router r* $\in$ *router_list* **do**
    router_bw $\leftarrow$ get_router_adv_bw(r)
    $B \leftarrow B +$ *router_bw*
    bw_list $\leftarrow$ *bw_list* $\cup$ *router_bw*
end
$C \leftarrow$ random_int(1,B)
while $T < C$ **do**
    $T \leftarrow T +$ *bw_list*$_i$
    $i \leftarrow i + 1$
end
return *router_list*$_i$

Figure 1: Pseudocode: Algorithm for Non-Entry Router Selection (source: [3]). B = combined bandwidth of all routers, T = the sum of bandwidths in the lower loop, C = randomly chosen number from B, i = number of the currently processed bandwidth in the lower loop, router_bw = advertised bandwidth currently processed in the upper loop, bw_list = list with all router bandwidths

After three routers have been selected randomly, the chain is formed by generating the encryption keys one by one with each router using *Diffie-Hellman-protocol*. The new key pairs are negotiated incrementally with each node after the keys have been established with the previous node in the path. This achieves so called forward-secrecy, meaning that any node cannot trace the keys other nodes are using to open the messages. After this procedure each router has its own encryption key for creating and peeling onion layers, and also information about who to send the messages to. Figure 2 shows the connection to the name service and the onion path after it has been formed.

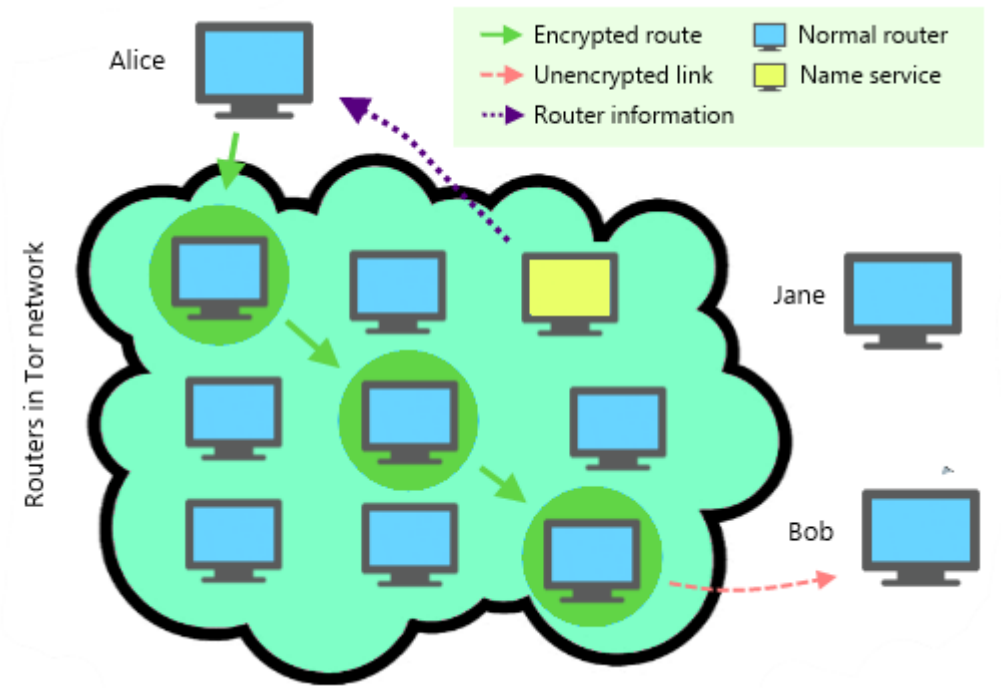

Figure 2: Alice's client downloads a list of onion routers from the name service and forms a circuit. The exit node sends the uncencrypted message to Bob.

As the result of forming the circuit separately with every router, one single node cannot know the addresses of all the other nodes in the circuit. The first router can see that the client is connected to the network, but it cannot see the final destination of the messages that the user sends over Tor. The entry node cannot know what the user is doing with the network. It receives messages and sends them forward in the chain. The third, final node can see where the messages are going to, but it cannot see who has sent them or what the data inside the message is. The third node called an *exit node*. The exit node sends the messages to the final receiver.

The final node in the circuit that receives the unencrypted message from the exit node is not necessarily aware that it is being part of an onion chain, because the messages leave the exit node unencrypted. Only by comparing the sender information from the header to a list of known Tor routers (a list is kept in the site torstatus.blutmagie.de [33]) the receiver can know that the messages are arriving from inside the onion network. Many internet services choose to block traffic coming through Tor addresses.

3.4 A message's journey through an onion circuit

DeFabbia-Kane [7] describes the course of a message through the onion network in a following manner: a client forms an *n*-length onion chain (as described previously), after which the chain consists of routers R_1, …, R_n to which the client has distributed a symmetric key K_i by using Diffie-Hellman-protocol. Before a message is sent, the client first encrypts it with an encryption key K_n, and after this with the key K_{n-1} all the way to level K_1.

Alice wants to send a message to Bob anonymously via Tor. She has formed an onion circuit with three onion nodes: $R_1 \rightarrow R_2 \rightarrow R_3$. The notation $[M]_{K_i}$ means that the message M is encrypted with a symmetric key K_i and the notation $[M]_{K_{i,j,k}}$ means that the message M is encrypted first with the key K_i, then with the key K_j, and after this with the key K_k. Before Alice sends the message through the circuit, her client encrypts it with the key K_3, then with the key K_2 and lastly with the key K_1. The message that Alice's client eventually sends will be $[M]_{K_{3,2,1}}$.

Every router opens, or decrypts, the message with their own encryption key K_i as the message travels inside the chain. After the router has decrypted the message, it sends it to the next router in the chain (or to Bob, if the router is the last node in the chain). As M travels through the chain, it will look like this:

$$Alice \xrightarrow{[M]_{K_{3,2,1}}} R_1 \xrightarrow{[M]_{K_{3,2}}} R_2 \xrightarrow{[M]_{K_3}} R_3 \xrightarrow{M} Bob$$

When Bob sends a response message M', he will send it to the router R_3, which encrypts it with the key K_3 and sends it backwards in the chain. Every router R_i encrypts the message with the key K_i. The path of the message backwards in the chain looks very similar to the path of the messages M path forwards in the chain. Because only Alice know all three keys K_1, K_2 and K_3, only she can open the message M' and read it.

$$Alice \xleftarrow{[M]_{K_{3,2,1}}} R_1 \xleftarrow{[M]_{K_{3,2}}} R_2 \xleftarrow{[M]_{K_3}} R_3 \xleftarrow{M} Bob$$

3.5 Cells

The messages that are passed between nodes inside onion circuits are called *cells* [8]. In addition to relaying data, cells can also be used for example for creating circuits (so called *CREATE cells*), destroying circuits (*DESTROY cells*) and keeping a connection alive (*PADDING cells*). Cells have a fixed size of 512 bytes mainly for the reason of preventing traffic analysis. All other cells but *relay cells* are called control cells and they have a header that describes their circuit identifier, which is called *circID*, and the purpose of the cell, called control cell command, or *CMD*.

The cells that are used for relaying messages are called relay cells. These cells have a little more information in the header than the control cells: in addition to circID and CMD the relay cells header states a stream identifier, a checksum and the length of the payload. The data payload is 498 bytes long and it can be used also to relay control cells to individual nodes in the circuits, for example when doing a Diffie-Hellman handshake. Figure 3 shows the structure of a control cell and a relay cell.

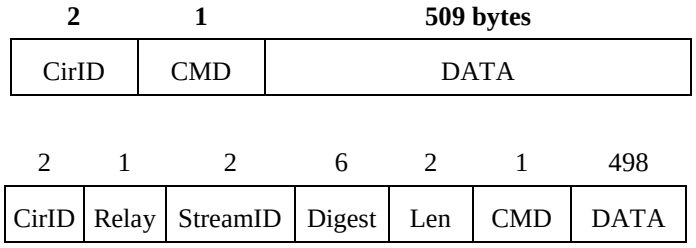

Figure 3: Control cell header (upper) and a relay cell header (lower) [8].

There are various types of relay cells. RELAY DATA cells are used to pass data forward or backward in the circuit. RELAY BEGIN cells are used to open a stream, which RELAY END cell closes. Relay cells are also used to create the actual circuit via RELAY EXTEND cells. The “extend” cell tells an onion router to pass a handshake on to a specified router and the router responds with a CREATED cell in case the handshake was successful. Figure 4 shows a diagram of how the cells are used when creating a two-hop circuit.

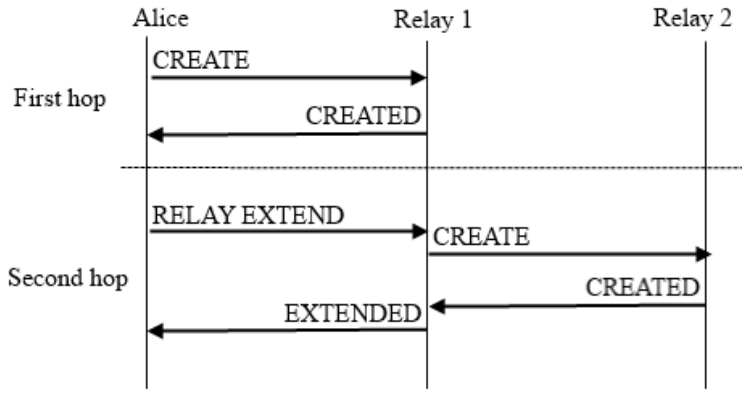

Figure 4: Creating the first two hops of a circuit.

4 Attack threats

The objective of onion routing is to prevent the localization and identification of the user. This means that attacks against the system try to connect for example logs of visiting a website or sending messages back to the user. In this chapter we take look at what kind of attacks have been developed against Tor. In what different ways can a potential attacker try to deanonymize users and how is the network designed to defend against these kind of attacks? Some attacks may have only the purpose of lowering the service quality with for example Denial-of-service attacks, but we leave these kinds of attacks out of our scope in this thesis and concentrate on deanonymizing attacks.

Onion routing protects against identification with obfuscation. In practice onion routing can only provide anonymity if it's user base is large enough [28]. The more users the network has, the more targets there are to combine a certain activity to. This means that a single user cannot establish their own onion network and create themselves anonymous protection. The strength of Tor is that it is not possible for outsiders to follow the path of the messages and also trace the messages that are coming out of the network back to whoever send them. This is at least the concept.

There are numerous different attacks devised against cryptographic systems. Most of the attacks against Tor are based on the attempt of taking over one or two routers within the network. Usually at least one entry node router within the circuit and possibly also the exit node within the same circuit. With this setup it is possible to do a traffic analysis attack that tries to associate inbound and outbound messages. *Passive attacks* monitor the stream of messages without interfering with it and *active attacks* modify the traffic somehow to make the analysis easier. In this chapter we will take a look at what other kinds of attack types are designed against the system.

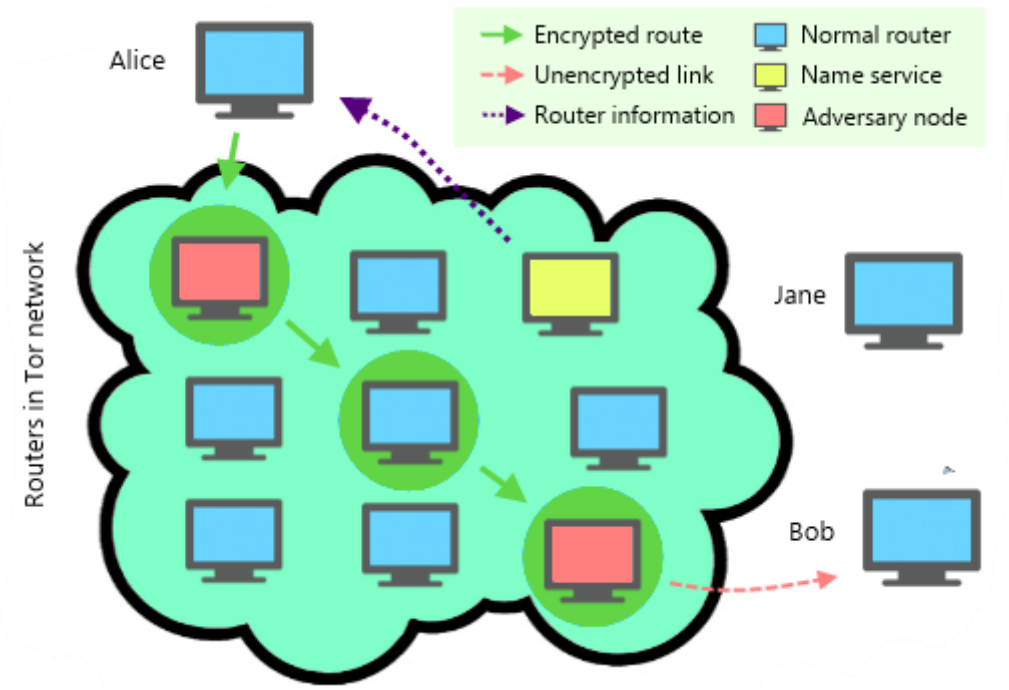

Figure 5: The adversary attempts to place the routers they control as an entry and exit node.

4.1 Attack categories

Salo [26] divides attacks into five categories according to different features. The five attack categories by Salo are *probabilistic models*, *entry and exit onion router selection attacks*, *AS and global level attacks*, *traffic and time analysis based attacks*, and *protocol vulnerabilities*. In his article Salo lists these categories and describes previously publicized attacks that belong to these categories. Let us go over these briefly.

Probabilistic models use mathematical probability models to analyse the network. These models are originally designed for Mix-networks and they use Bayesian probability models to measure how secure the network is. The same kind of models have been designed for Tor [11]. The models treat the system as a black box, trying to create mathematical models based on two assumptions: firstly, that only one user is linked to one output and secondly, that the output can be linked back to the user in case both can be observed. These models are not actually attacks, but rather offer a way of measuring how possible it is for an attacker to deanonymize a user.

Entry and exit onion router selection attacks aim at increasing the probability of the attacker's routers to be chosen as the entry and/or exit node. Salo mentions two attacks from this category: first, *Compromising Anonymity Using Packet Spinning* [24], tries to create loops between routers so that they can be made to refuse serving other clients, in practice killing the router. The goal is to inflate the probability of the malicious routers to be chosen to be a part of a particular circuit. Another attack in this category is *Low-Resource Routing Attacks Against Tor* [5] that tries to announce false information about the bandwidth and uptime of the router to the name service, which gives the router more priority when entry nodes are chosen.

Traffic and time analysis based attacks covers the biggest amount of attacks as a category. These kind of attacks try to weaken anonymity by observing patterns in the traffic, for example in network flow timing. With these patterns the adversary can correlate incoming and outgoing network traffic. Passive attacks observe the packets going through the system and active attacks try to for example water mark packets to make them easier to analyse.

There are seven research papers in this category: *Low-Cost Traffic Analysis of Tor* [19] presents a way to trace nodes that are used in a circuit and also a way to link unrelated streams to their initiator based on different latencies in streams, *A cell counter based attack against Tor* [16] presents traffic analysis technique by water-marking the cell counter of a cell, *Browser-Based Attacks on Tor* [1] uses a man-in-the-middle attack and modifies HTTP-traffic to improve analysis, *A Practical Congestion Attack on Tor Using Long Paths* [10] uses a congestion attack with HTTP-stream modification to learn circuit paths, *How Much Anonymity does Network Latency Leak?* [12] measures network latencies to recognize a user, and *Passive-Logging Attacks Against Anonymous Communications Systems* [40] observes user behaviour and tries to predict the initiator of a certain path.

AS (autonomous system) and global level attacks assume an attacker that can observe or manipulate a large part of traffic that goes in and out of the network. This kind of attacker can use for example traffic and time analysis attacks to recognize data flows. Tor is not designed to protect the user against *AS* (meaning *autonomous system*) -level adversary [28]. Salo mentions two papers *AS-awareness in Tor Path Selection* [9] that tries to evaluate how likely an AS level attacker actually is, and

Large Scale Simulation of Tor [21] that evaluates several traffic analysis attacks in a simulated network.

Protocol vulnerabilities is the last category. These attacks try to find weaknesses from communications protocols. *Effective Attacks in the Tor Authentication Protocol* [41] suggests a vulnerability in the way session keys are distributed, that will lead to inconsistencies in the session keys. *On the risks of serving whenever you surf* [18] tries to recognize bridges in the Tor-network. The attacker first tries to recognize possible bridge-candidates and after this tries to confirm the findings with an *active circuit clogging attack*.

5 New attacks against Tor

Attacks against Tor seems to be a popular research subject at least by looking at the amount of papers one can find with a simple google scholar search. After 2010 there has been a lot of new papers on the subject and we will look at some more popular ones and less popular ones here.

5.1 The Bad Apple Attack

In 2010 researchers Le Blond et al. from French Institute for Research in Computer Science and Automation discovered a way of tracing TCP streams of *peer-to-peer* (P2P) applications running over Tor [4]. With this attack the researchers were able to trace nine percent of all streams going through their exit nodes, revealing among other things 10,000 BitTorrent users, which is a protocol for sharing files in a P2P network. With this information they were able to profile BitTorrent usage.

5.1.1 Outline of the attack

The researchers conducted their experiment over 23 days, monitoring six exit nodes that they controlled. For ethical reasons they did not store any information that could be used to identify the targets later. The attack is based on the observation that although BitTorrent users may search for peers from inside Tor, actually majority of them still do the actual sharing of the content outside of the Tor network. There may be multiple reasons for this: firstly, sharing content outside of Tor is simply much more efficient due to higher bandwidth, and secondly Tor does not support UDP, which is the protocol that BitTorrent uses to share information about users in the P2P network with so called DHT trackers.

The researchers found that 72 percent of BitTorrent users either subscribe their public IP to the DHT tracker and/or form P2P connections with their public IP. The attacker can easily add their IP to the list of peers and wait until another peer connects to it. By comparing the connected IP to the list of known Tor exit nodes it is possible to know whether the connection is coming from Tor or outside of it. Another way of identifying users is by monitoring subscriptions to the DHT through an exit node.

The subscribed IP must be outside of Tor, because Tor does not support UDP used by the DHT. By comparing the subscribed IP/port combinations in the list of subscriptions and handshakes to the targeted user, it is possible to identify the Tor user.

After revealing the source IP of one stream in an exit node, all streams within that circuit can be traced to the same user. Tracing streams in different circuits can be done with the peer identifier or by comparing IP/port. With this information the researchers were able to profile BitTorrent usage inside Tor, for example amount of streams, countries of origin and amount of users. The analysis also revealed huge amount of other streams, such as HTTP streams from some browsers. With the browser information the researchers were able to see what kind of websites normal Tor users were interested in.

The original article [4] adds to other studies trying to profile Tor users, but it also introduces a new attack method that could be used to reveal IP addresses of the users. This could also potentially threaten other anonymity offering services. Defences against the attack mostly depend on the user to take caution when using the system. The researchers also note that Peer Exchange (PEX) and centralized UDP tracking could still be exploited in same way, which could be a potential research subject in the future.

5.2 Application-level attack

A paper published in 2011 by researchers from Southeast University of China and Changzhou College of Information Technology China, introduces a man-in-the-middle attack against TCP based low-latency applications [38]. There are two attack schemes: in *forged webpage injection attack* a malicious exit node responds with fake web pages to the client's web requests, creating a distinctive traffic pattern for traffic analysis. *Target webpage modification attack* is similar, but instead than responding with a totally new webpage, it passes the original page forward to minimize lag, but instead injects invisible links in to the page. Both of these attacks can potentially be used to identify a client.

5.2.1 Outline of the attack

The first scheme, forged webpage injection attack, creates a more distinctive pattern than the second one, so recognizing flows is more effective. The attack requires one entry router and one exit router to be placed in to the Tor network. Exit router responds to a web request with a forged webpage that contains *img* tags that the target's web browser is then tricked to fetch. Example of a forged webpage from the original paper [38] is shown below. The request to fetch these images creates a traffic pattern consisting of relay cells five times the amount of the number of inserted *img* tags within a specific timespan. The entry router can recognize the cells from the circuit identifier (CircID) and cell command (CMD or Relay). After this the communication flow can be confirmed by comparing pattern observations between the entry and exit router.

```
<html>
<head>
<meta http-equiv="refresh" content="w;url=URL_Bob">
</head>
<body>
<img src="URL_Bob/img1.gif" style="display:none"/>
<img src="URL_Bob/img2.gif" style="display:none"/>
<img src="URL_Bob/img3.gif" style="display:none"/>
… …
<img src="URL_Bob/imgm1.gif" style="display:none"/>
</body>
</html>
```

Source code 1: Example of a forged webpage. Source: [38]

The first version of the attack can be detectable because it creates a delay in the communication. The second scheme, *target webpage modification attack*, works similarly to the first scheme, but the difference is that the exit node responds with the authentic web page that it injects some *img* tags into. In this situation the traffic pattern is harder to detect because the webpage can cause some additional traffic. Launching the attack several times improves the accuracy.

5.2.2 Results and defenses

The researchers analysed the attack and also conducted some experiments with a small sample network. According to the article [38], the analysis shows both schemes of the attack to be viable for profiling clients and hidden servers. Detection rate of scheme one does not suffer from links within the target webpage, but the detection rate of scheme two decreases if the target webpage contains links. Still both schemes remained effective for correlating the connection.

Three defences are proposed against the attack. First defence is to reduce the change of selecting a malicious router to the circuit by deploying more unmalicious routers to the network and by improving the router selection algorithm to be stricter with the selection. Second defence is to try to detect abnormal activity such as fetching invisible images before showing a webpage. Third defence is to use HTTPS for connection to prevent man-in-the-middle attack.

5.3 Probabilistic Analysis in a Black-box Model

Joan Feigenbaum from Yale University, and Aaron Johnson and Paul Syverson from U.S. Naval Research Laboratory published a paper in 2012 about probabilistic analysis of onion routing [11]. This method is not actually an attack against the system in the definition that it would try to strip away the user's anonymity, but it does fall under the attack categories we introduced in the beginning of this chapter. The purpose of this article is to provide a mathematical proof of the anonymity that onion routing provides. The proof is modelled against a computationally bounded active adversary that compromises some unknown part of the system.

The model that they use for the analysis is an abstraction of an anonymous-providing system, so it could potentially also be used to analyse other anonymous systems. There are three assumptions made about the system in the model they describe: Firstly, it is possible for the adversary to know when they are observing a user or not. Secondly, a single user can be linked only to one connection in the system. The third assumption is that it is not possible for the adversary to know whether a user and a connection are part of a same flow just by observing the system.

The anonymity that this model describes is the *relationship anonymity* of users, measuring how well an adversary can determine if two parties have communicated with each other. Basically the anonymity of the system is measured with the posterior probability of user choosing a certain destination. In their 29-page paper the researchers estimate simplistic lower bound values and worst case upped bound values to the user's anonymity in Tor network.

The paper shows that an user's anonymity is the worst when at least every other user has the same destination as them, but this is of course a very unlikely situation and this is why a more realistic scenario based on Zipfian distribution is evaluated. (Zipf's law tells us that the connections to destinations are not evenly distributed, but the most popular connections will have exponentially more connections than the less popular ones.) By using Zipfian distribution it is shown that the anonymity of the system is usually very strong, close to the lower bound.

This model is of course an abstraction that does not take into account many known attacks against onion routing, but it is trying to continue an on-going process of giving a formalization of anonymity that covers most of these situations. More work studying for example the effect of entry node and entry guard security is a subject for future research.

5.4 CellFlood

CellFlood is an attack that can be used to render Tor routers unavailable for users by sending requests continuously to them until the router refuses to serve other clients [2]. This Denial-of-Service attack was discovered in 2013 by four researchers from Sapienza University in Rome and Columbia University in New York. The researchers also developed a defence including so-called *client puzzles* against this attack, which also was added to the Tor software.

5.4.1 Outline of the attack

The attack makes use of the fact that encrypting messages with public keys is about twenty times easier than opening the encrypted messages with a private key. When the onion circuit is first created, the session keys for each router are distributed using

each router's public key by using so called CREATE cells. The cost of sending copious amounts of CREATE cells to an onion router is a lot cheaper for an attacker than it is for the attacked router.

The router processes the CREATE cells in a separate process from the main relay thread, so receiving large amounts of CREATE cells does not disturb the message relay process. However, the background process for creating new connections can only do a limited amount of work, and when it cannot process any new requests, it starts discarding them. It was discovered that the adversary router does not even need to create new cells for every request it sends, but it can keep sending the same cell over and over again. This makes it possible to perform a cheap DoS attack to prevent a certain router from creating any new circuits and serving any new clients.

5.4.2 Results and defenses

The researchers experimented with the attack in a controlled environment and with the real-world Tor network and in all of their tests the attack remained very effective in disturbing the targets capability of processing new requests [2]. They also estimated how much of an impact CellFlood could have and estimated that an adversary would need little bandwidth, around 116-232 Mbits/s, to clog a single router.

The proposed and implemented defence is to make the client do some computationally intensive work every time it wants to send a processing request to a router. This work usually involves solving a cryptographic puzzle that the entry router verifies to be correct. The researchers measured that this kind of client puzzle takes at most moderate amount of toll on genuine clients and is effective at stopping CellFlood – type DoS attacks.

5.5 EgotisticalGiraffe

In the year 2013 Edward Snowden famously leaked thousands of top-secret documents of the *National Security Agency* (NSA) in United States. One of these thousands of documents was a PowerPoint presentation that described how the agency was able to identify some Tor users (published for example here: [25]). Although

there is no official documentation available for this attack method, it is easy to get a picture of how it was done, so we will take a brief look at the method here.

5.5.1 Outline of the attack

The attack has two phases: the first aim is to identify Tor users in the internet and after this the aim is to direct the identified users to a site that would infect the user's computer with some custom malware for further identification. The first phase of identifying a Tor user is fairly straightforward. The package in which users download the Tor application software is called *Tor Browser Bundle* and it comes with a custom Firefox browser configured for browsing Tor. Usually Firefox browsers announce a *BuildID* that tells a website the release version of the browser that a user is browsing the site with. The slides announce that Tor Browser Bundle's BuildID was by default set to zero, instead of a standard build number, meaning that users accessing sites with Firefox browsers with BuildID zero are potential Tor users.

The second phase of the attack is to do a man-in-the-middle attack to infect the potential Tor user's computer with a malicious Firefox exploit. The target user was directed into a special server that would then infect the user's browser with a specially designed exploit. The particular exploit mentioned in the leaked slides uses an extension for JavaScript, but it is rumoured that also other vulnerabilities were used [27]. After the user's computer was infected with this malicious exploit, it would send all kinds of callback data about the user, which could easily be used to identify them.

5.5.2 Results and defenses

The slides also report that some tests were conducted and there were some initial problems with Tor Browser Bundle but eventually they got it working. Other than that it is not known how much user data this particular attack was able to produce. Shortly after this attack was publicized the JavaScript vulnerability in Firefox was fixed by Mozilla by disabling the exploited feature called *E4X*. Along with this also the vulnerability in the Tor Browser bundle was fixed. The Tor Browser Bundle also comes with an option called *NoScript* that would disable this kind of attacks but it is not enabled by default and requires the user to enable it by themselves.

5.6 Sniper Attack

In 2014 four researchers from NRL and Humboldt University of Berlin found a new weakness in Tor's flow control algorithms [13]. They named a new attack exploiting this weakness *Sniper Attack*. The attack requires minimal amount of resources and it can be used to paralyze arbitrary nodes from the Tor network. According to the researchers, Sniper Attack can be potentially used to break the anonymity of hidden services. They developed three different defences against this attack and one of these defences were added to the Tor software.

5.6.1 Outline of the attack

Sniper Attack exploits Tor's so-called flow control. Flow control is used in the situation when large amounts of packets are sent from one location to another, for example from server to client. Just sending a big chunk of data packages at once may overload the receiving end so flow control is needed to pace the rate of package delivery. Flow control allows the client to control the rate of transmission by making the exit node of the circuit to buffer data and allowing the client to send so called *SENDME* cells every time it is ready to read from this buffer. The delivery end keeps a buffer of 1000 cells in its memory and sends 100 cells for every SENDME call it receives (adding another 100 cells to the buffer). The protocol goes as follows:

1. Client has formed a circuit through onion network to an external server
2. **Client:** Send request to start downloading data from the server
3. **Exit:** Sends the request to server and starts buffering data up to a 1000 cells
4. **Client:** Reads data and sends SENDME when 100 cells have been read
5. **Exit:** Adds another 100 cells to the buffer
6. If not all data have been read, go to step four

This protocol can be exploited to attack either entry nodes or exit nodes by two different ways. The first version of the attack is illustrated in figure 5. If a malicious party wants to shut down an entry node, it forms a circuit in such a way that the adversary's own node is used as an exit. The protocol assumes that the exit node does the actual flow control, but because in this situation the exit is being controlled by the attacker, it can send as much data as they want to, ignoring the package flow con-

trol limits. The trick here is that the client does not read from the buffer and the exit sends loads of data forwards, which causes the data packets to be built up to the entry node's buffer, eventually causing it to crash.

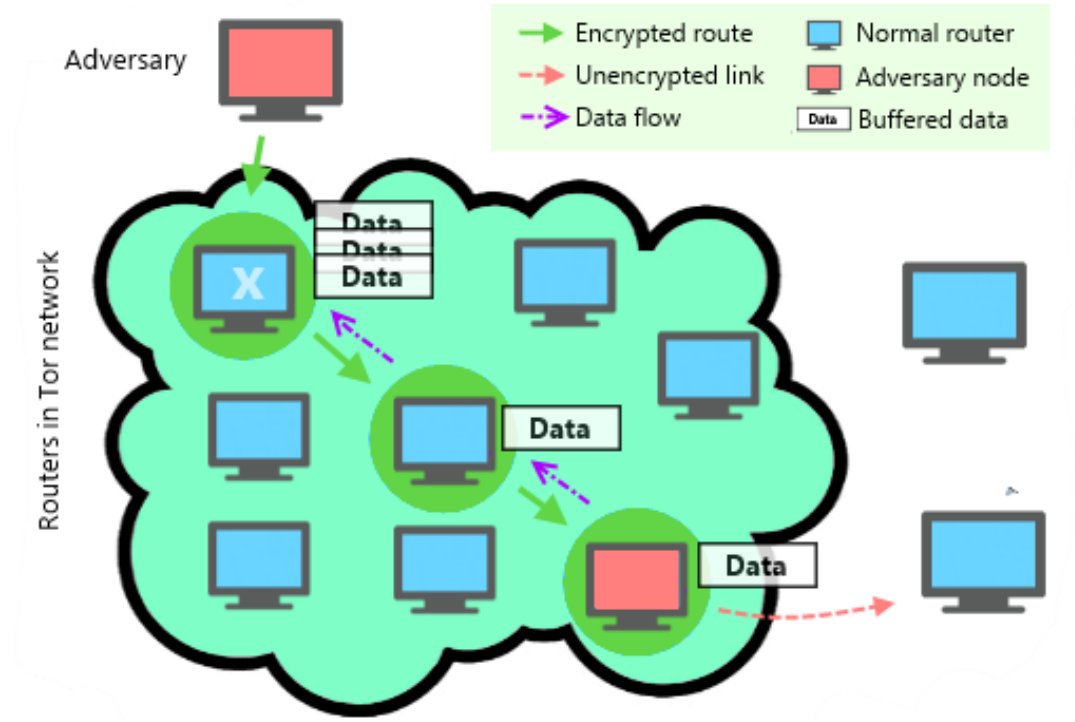

Figure 6: Adversary controls the client and the exit node, targeting the entry node (marked with X). The exit node sends amount of data through the circuit and simultaneously client stops reading from the entry node which causes the entry nodes buffer to fill up and eventually crash. [13]

The other way of targeting exit nodes goes very similarly, but the data is sent to the other direction in the circuit. The client, controlled by the attacker, sends loads of data towards a server while again ignoring the flow control limits. The receiving server, owned by also by the adversary, stops reading from the connection, causing the data to fill up the exit nodes buffer. This also causes the targeted node to eventually crash.

There is also an efficient version of the second attack, which requires the adversary to only control a client. The exit node has the 1000 cell limit, after which it stops buffering data and closes the circuit, but the middle and entry nodes do not have such limits, and they will keep on buffering until the client instructs otherwise. Keeping the exit node sending packages by sending SENDMEs, the adversary can keep injecting data to the circuit without reading from it. By combining multiple streams the attack can be very effective. Sniper attack, including how it can be performed while

avoiding detection, is described in more detail in the paper *The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network* [13].

5.6.2 Results and defenses

Sniper Attack can be used to force a hidden service to choose the attackers node as a guard relay and to use this malicious guard relay to deanonymize the hidden service. The researchers tested this attack and found out that deanonymizing hidden services with this attack would be possible and estimate that attacker could use it to disable top 20 Tor exit relays.

The defence that was chosen to be added to Tor is to kill a circuit if its memory starts to run out. After this the Sniper attack became ineffective. The authors however remind that even though the Sniper attack cannot no longer be used to DoS a router, it can be still used to consume its bandwidth. [20]

5.7 New traffic confirmation attacks

During the recent years there have been published a number of traffic confirmation (or traffic correlation) attacks using different techniques, for example flow records from a software called Netflow [5], signal injection to pass information between nodes [32] and circuit fingerprinting by tagging the traffic to make correlation easier [14]. Traffic confirmation attacks as a group has definitely the biggest number of attacks developed. The Tor development team has from the beginning, commented on traffic confirmation attacks, saying that it is trivial to focus on traffic confirmation attacks, since onion routing does not try to defend against these. Instead Tor tries to defend against traffic analysis attacks, which the adversary can use to know which points in the network to attack [8].

In 2014 the developers commented on their blog about the new traffic confirmation attack on using netflow logs that:

> "It's great to see more research on traffic correlation attacks, especially on attacks that don't need to see the whole flow on each side. But it's also important to realize that traffic correlation attacks are not a new area". [37]

We will go through some of the new attacks briefly.

5.7.1 Flow records

A paper titled *On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records* (2014) evaluates if NetFlow -network analysis software packaged within Cisco routers can be used to analyse anonymous networks such as Tor [5]. The NetFlow software itself is not accurate enough for traffic analysis, but the records it provides can potentially be analysed to find a source of a connection coming to a server. The authors present an attack that assumes a large AS-level adversary, but they say that the attack can also be done by a non-AS attacker if a target entry node can be identified.

The attack uses NetFlow data between the server and the exit, and entry and the client to find matched patterns with Pearson's correlation coefficient. It also injects some traffic patterns in to the TCP connection. The paper reports 81,4 % success rate with 12,2 % false negatives and 6,4 % false positives on the real Tor network.

5.7.2 "Relay early" Traffic confirmation attack

This is another attack that has no paper written about, but got a lot of publicity [34] [29]. The details were reported in the Tor blog in July 2014 when it was detected that some unknown attacker was doing an active traffic confirmation attack against Tor network [32]. The attack exploited so called "relay early" –cells that were added to Tor to prevent building very long paths. The attackers used these relay early cells against their designed purpose to send information about hidden service addresses from exit nodes to malicious entry nodes which the attackers has previously placed into the network.

It is not known what information the attackers were able to gain from this attack or even if the attackers purpose was research, mischief or something else. Most likely the attackers did not get any user or hidden service addresses or anything else that would compromise user security [32]. As usual the Tor software was upgraded with defences, main one preventing "relay early" cells to be used in attacks like this. To prevent this kind of attacks in the future there is also a plan to change Tor clients to

use only one entry guard, instead of three, to reduce the exposure of clients in the network, but this has not yet been implemented.

5.7.3 Circuit fingerprinting attacks

The paper *Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services* published in 2015 takes a look at some possible weaknesses in hidden service communication and proposes two new attacks with some possible defence schemes [14]. The first attack starts by looking for traffic that looks like hidden service creation and after this tries to identify the service with a *website fingerprinting* attack. The second attack is aimed at the scenario where the client uses only one entry guard, in which case a malicious entry guard can see all of the client's circuits.

The effectiveness of both attacks were evaluated in the real-world Tor network. The researchers report 98% and 99% present true positive rates with the attacks and 0.1 % and 0.07 % false positive rates. They also propose some defences against the attacks: reducing circuit lifetime and sending random padding cells. Against the second attack it is suggested to build circuits pre-emptively to eliminate the identification of hidden service circuit establishment.

5.8 A Stealthy Attack Against Tor Guard Selection

In 2015 three researchers from University of Electronic Science and Technology of China and Chinese Academy of Sciences discovered a way to speed up the rate at which new guard nodes are chosen to an onion circuit in a situation where an attacker is able to control part of user's entry point traffic [17]. Normally the client uses a single guard node for one to two months before choosing another one. The probability of choosing a malicious entry node is described in the following way:

> "…if an attacker controls C out N relays (ignoring bandwidth), then the attacker will control both the entry and exit nodes of any given circuit with probability $(C / N)^2$ … While using entry guards, we would have probability (N - C) / N – to choose a good entry and not be compromised until the next round of entry guards " [17]

5.8.1 Outline of the attack and results

The first part of the attack aims to increase the possibility of a target user (called Alice) selecting the attackers router as the guard node. The guard nodes are selected randomly, but more weight is given to nodes with higher bandwidth. This is why the attacker needs to deploy routers with a high bandwidth as guard nodes. Also, the more guard nodes the attacker has in their control, the more likely it is for Alice to select one of these as a router. So increasing the change of Alice selecting a malicious guard node is done by deploying as many high bandwidth guard nodes as possible.

The second part of the attack tries to shorten the time in which Alice selects the guard nodes. This time interval is normally 30-60 days, but it can be increased to around one to one and a half minutes by blocking two out of the three selected guard nodes Alice has selected. As this attack requires the attacker to be able to control part the target user's traffic, they can block the traffic to all but one guard nodes.

The researchers evaluate the attack to be very hard to detect and very effectively compromising the target user's anonymity. The attack does not affect the targeted user's network usage and their experiment results show that more than 80% of the users can be compromised in about 30 minutes with this technique.

6 Refined attack categories

After taking a look at some papers published in the recent years, we now have a wider picture of different ways with which an adversary could approach the task of breaking the anonymity provided by onion routing. The list of papers mentioned in this chapter is certainly not exhaustive, but the purpose is rather to provide examples of different types of existing attack techniques. After looking at theoretical and practical examples, it is easier to classify older and newer attacks and see where what their unique characteristics are.

We have created some categories in order to classify the attacks that are introduced previously. The classification criteria has mainly been the main purpose of the attack and what kind of features it exploits. Often the categorization is not clear. The methods may have one or more distinctive features that may be similar to two different categories, the attack may have multiple different uses or different types of attacks may be chained together, which make it harder to differentiate the category. Looking at the designed main use however often aids with this process and we have been able to create distinct categories, which we will introduce next.

In our classification we have dropped “probabilistic models” from our previously introduced categories in the beginning of this chapter and combined “protocol vulnerabilities” with other categories. The reason that probabilistic models are not included in our classification is that they do not belong under our main topic since these kind of techniques do not pose any kind of methodologies or vulnerabilities that help with compromising the anonymity of Tor. These models aim at evaluating how good the security of Tor is, not to “break it”.

The category “protocol vulnerabilities” can be a subcategory of either “entry and exit onion router selection attacks” or “traffic and time analysis based attacks” and sometimes the distinction between these categories is blurry, so we do not separate attacks using protocol vulnerabilities. In some cases the distinction however can be made. Some of the attacks falling under protocol vulnerabilities can be used for other kinds of mischief, such as lowering the usability of the network, but here we will only con-

centrate on the aspect of an adversary trying to deanonymize users or hidden services.

6.1 Final categories

Our first proposed category is **Entry and exit onion router selection attacks**. These attacks raise the possibility of the adversary's router to be selected as an entry and/or exit node to a circuit. The usual way is to do some kind of Denial-Of-Service attack to stop other routers from serving other clients, but there are also other ways to reach the same goal. For example [13] and [2] belong to this category, because they try to consume resources of non-malicious routers to get the adversary routers to be chosen as an entry or exit node.

The next category, **Traffic and time analysis based attacks,** covers attacks that try to analyse traffic and try to correlate streams to connect circuit source to its destination. This category can be divided into passive and active attacks. Passive attacks monitor the traffic without altering it and active attacks can use for example message tagging to make the analysis more efficient. It is easy to see that for example the traffic confirmation methods introduced in [32] and [14] fall under the category of traffic and time analysis based attacks.

We have given the attacks introduced in [11], [38] and [25] their own category because they have very distinct characteristics from any other attack category. We first thought about naming this category "application-level attacks", but because in the OSI-model [23] "application-level" refers to the TCP/IP layer that Tor is part of, to avoid confusion we suggest the term **software-level attacks** for this category. Attacks in the category "software-level attacks" do not utilize the Tor implementation or protocols directly, but they rather make use of *the applications that use Tor connection to download and send data*. Usually these types of applications are not designed with anonymous protocols in mind and this is why they may leak some sensitive information about their users. For example application-level attack in [38] makes use of the fact that normal HTTP-level protocols are not designed against traffic analysis. With this technique the browser can be used to create distinctive traffic patterns by sending it some, normally harmless, HTML-code.

The fourth category is **AS and global level attacks**. Attacks in this category require substantially more resources. They assume a powerful adversary that can observe and/or manipulate a significant part of the traffic going in and out of the whole network. Usually attacks in this category can only be evaluated on a theoretical level or with simulations. The method described in [5] utilizes a traffic analysis attack which could also be used by non-AS-level adversary, but it would need an AS-level attacker to be truly effective. The attack that blocks some connections described in [17] could not be executed unless the adversary could manipulate a large part of all the network connections.

We have described our attack categories and what kind of attacks fall under them and next we will give an overview of the final categories. We propose the following attack categories and list the related papers under them. The papers from previous categorization that fall under our classification categories are also mentioned, with the old category marked in (brackets):

Software-level vulnerabilities

- One bad apple spoils the bunch: exploiting P2P applications to trace and profile Tor users [4]
- A potential HTTP-based application-level attack against Tor. Future Generation Computer Systems [38]
- EgotisticalGiraffe [25]
- Browser-Based Attacks on Tor [1] (Traffic and time analysis based attacks)

Traffic and time analysis based attacks

- "Relay early" Traffic confirmation attack [32]
- Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services [14]
- Low-Cost Traffic Analysis of Tor [19] (Traffic and time analysis based attacks)
- A cell counter based attack against Tor [16] (Traffic and time analysis based attacks)

- A Practical Congestion Attack on Tor Using Long Paths [10] (Traffic and time analysis based attacks)
- How Much Anonymity does Network Latency Leak? [12] (Traffic and time analysis based attacks)
- Passive-Logging Attacks Against Anonymous Communications Systems [40] (Traffic and time analysis based attacks)
- On the risks of serving whenever you surf [18] (Protocol vulnerabilities)

Entry and exit onion router selection attacks

- The sniper attack: Anonymously deanonymizing and disabling the Tor network [13]
- CellFlood: Attacking Tor onion routers on the cheap [2]
- Compromising Anonymity Using Packet Spinning [24] (Entry and exit onion router selection attacks)
- Low-Resource Routing Attacks Against Tor [5] (Entry and exit onion router selection attacks)

AS and global level attacks

- A Stealthy Attack Against Tor Guard Selection [17]
- On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Record [5]

6.2 Threat model

Based on the picture we have provided in this chapter, we see that a theoretical attacker would have a certain range of options with which to start deanonymize users or hidden services. A so called global adversary could use a traffic correlation attack to identify a large number of flows so they would not have to do anything in addition in order to get a hold of traffic information. However, as [17] describes, they could also use some kind of Entry and exit router selection attack to improve their position for traffic analysis. However this may not be necessary.

In case of a non-AS-level attacker, the main attack class that the attacker would use to break the anonymity, is again traffic analysis. However, before being able to conduct any traffic analysis, they would have to do some preliminary steps in order to get into a position of being able to use a traffic confirmation attack. We have seen that the attacker can use two different ways to get this to this position: entry and exit onion router selection attacks or software-level vulnerabilities.

Entry and exit onion router selection attacks cannot by themselves be used to deanonymize users, so they need to always be followed by some kind of passive or active analysis method. Software vulnerabilities however may reveal some information about users, so they may also lead to deanonymization directly. Figure 5 gives a visualization of the attack model.

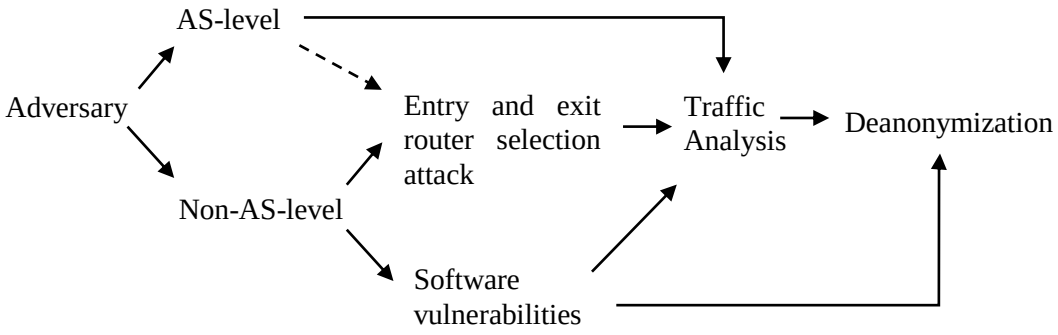

Figure 5: The adversary has various options to reach the goal of deanonymizing the target.

7 Conclusion

In the first part of this thesis we have travelled through the history of onion routing from the first version of onion routing published in 1996 to the Tor-application of today. What started as a US navy project is now carried on by an organization called *The Tor Project* and funded by Electronic Frontier Foundation. During this history three generations have been published from which the last one is still under active development as can be seen for example from the large amount of research papers that have been published in the recent years.

Onion routing is based on a simple idea about how to provide anonymization to users through a distributed solution and we have taken a look at this technology in the second part of this paper. The Tor network is the largest known implementation of onion routing with 7000 routers relaying messages through the network for its 2 000 000 clients. Clients create connections to servers via circuits consisting of three routers as nodes and the onion –data structure is used to transport data through these circuits. Tor gives its users so called *relationship anonymity*, meaning that the communication relationship between two parties is hidden to all but the communicating parties themselves.

The last part of the paper goes a bit deeper into the information security of Tor. We first take closer a look into eight papers published since 2010. The papers are chosen on basis of providing examples of different types of attacks against the anonymity of the Tor user. After this we divide these attack papers into categories according to what is the purpose and exploitation method of the attack. The categories we propose are: Entry and exit onion router selection attacks, Traffic and time analysis based attacks, AS and global level attacks, and Software-level attacks.

7.1 Thoughts about the security of Tor

Onion routing is still being developed actively and any new security flaws are being fixed once they are discovered, but it seems that there have been no great revelations in the security of Tor from the first days of the technology. It has been shown that onion routing is able to withstand large part of the attacks and any successful attacks

mostly rely on vulnerabilities outside of the Tor protocol such as user mistakes or JavaScript vulnerabilities. The developers have from the beginning stated that Tor is vulnerable against a global passive adversary, but is this really true?

The principle of making Tor a low-latency service means that there is a constant adjustment of balance between security and usability. The low-latency principle means that in theory Tor does not have security against a global passive adversary doing traffic correlation. Adding Mixing, padding or some other technology would make the onion technology so expensive to run, that it would be unusable.

However, the principles of making the software open source and allowing free riding lower the bar for an user to adapt the technology. Because of this Tor has been able to attract a large amount of users which makes onion routing still very resistant against traffic correlation. The main problem with traffic correlation attacks is that they use machine learning which means that in order to reach high confidence levels they will require large amounts of data. Even a small false positive rate, such as 0.1% means tens of thousands of falsely flagged streams in a sample of millions of streams and this can be hard to overcome.

The evaluation still continues of whether or not an AS really exists that could monitor large enough part of the Tor network to *confirm* the relationship between flows [11]. As the size of the network grows, the more resources a global adversary would require to monitor a large enough part of the traffic to do the correlation. Tor is an open project and the developers encourage users to run relays in their computers to increase the security of the network and also promote researchers to continue searching for possible security flaws from the technology [31]. Only the future will show whether the Tor network can serve its increasingly large user base with low latency and high security.

References

[1] Abbott, T. G., Lai, K. J., Lieberman, M. R., & Price, E. C. (2007, June). Browser-based attacks on Tor. In Privacy Enhancing Technologies (pp. 184-199). Springer Berlin Heidelberg.

[2] Barbera, M. V., Kemerlis, V. P., Pappas, V., & Keromytis, A. D. (2013). CellFlood: Attacking Tor onion routers on the cheap. In Computer Security–ESORICS 2013 (pp. 664-681). Springer Berlin Heidelberg.

[3] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., & Sicker, D. (2007, October). Low-resource routing attacks against tor. In Proceedings of the 2007 ACM workshop on Privacy in electronic society (pp. 11-20). ACM.

[4] Blond, S. L., Manils, P., Abdelberi, C., Kaafar, M. A. D., Castelluccia, C., Legout, A., & Dabbous, W. (2011). One bad apple spoils the bunch: exploiting P2P applications to trace and profile Tor users. arXiv preprint arXiv:1103.1518.

[5] Chakravarty, S., Barbera, M. V., Portokalidis, G., Polychronakis, M., & Keromytis, A. D. (2014, March). On the effectiveness of traffic analysis against anonymity networks using flow records. In Passive and Active Measurement(pp. 247-257). Springer International Publishing.

[6] Chakravarty, S., Barbera, M. V., Portokalidis, G., Polychronakis, M., & Keromytis, A. D. (2014, March). On the effectiveness of traffic analysis against anonymity networks using flow records. In Passive and Active Measurement(pp. 247-257). Springer International Publishing.

[7] DeFabbia-Kane, S. (2011). Analyzing the effectiveness of passive correlation attacks on the tor anonymity network (Doctoral dissertation, Wesleyan University).

[8] Dingledine, R., Mathewson, N., & Syverson, P. (2004). Tor: The second-generation onion router. Naval Research Lab Washington DC.

[9] Edman, M., & Syverson, P. (2009, November). AS-awareness in Tor path selection. In Proceedings of the 16th ACM conference on Computer and communications security (pp. 380-389). ACM.

[10] Evans, N. S., Dingledine, R., & Grothoff, C. (2009, August). A Practical Congestion Attack on Tor Using Long Paths. In USENIX Security Symposium(pp. 33-50).

[11] Feigenbaum, J., Johnson, A., & Syverson, P. (2012). Probabilistic analysis of onion routing in a black-box model. ACM Transactions on Information and System Security (TISSEC), 15(3), 14.

[12] Hopper, N., Vasserman, E. Y., & Chan-Tin, E. (2010). How much anonymity does network latency leak?. ACM Transactions on Information and System Security (TISSEC), 13(2), 13.

[13] Jansen, R., Tschorsch, F., Johnson, A., & Scheuermann, B. (2014). The sniper attack: Anonymously deanonymizing and disabling the Tor network. OFFICE OF NAVAL RESEARCH ARLINGTON VA.

[14] Kwon, A., AlSabah, M., Lazar, D., Dacier, M., & Devadas, S. (2015). Circuit fingerprinting attacks: passive deanonymization of tor hidden services. In 24th USENIX Security Symposium (USENIX Security 15) (pp. 287-302).

[15] Kwon, A., AlSabah, M., Lazar, D., Dacier, M., & Devadas, S. (2015). Circuit fingerprinting attacks: passive deanonymization of tor hidden services. In 24th USENIX Security Symposium (USENIX Security 15) (pp. 287-302).

[16] Ling, Z., Luo, J., Yu, W., Fu, X., Xuan, D., & Jia, W. (2009, November). A new cell counter based attack against tor. In Proceedings of the 16th ACM conference on Computer and communications security (pp. 578-589). ACM.

[17] Li, Q., Liu, P., & Qin, Z. (2015). A Stealthy Attack Against Tor Guard Selection.

[18] McLachlan, J., & Hopper, N. (2009, November). On the risks of serving whenever you surf: vulnerabilities in Tor's blocking resistance design. InProceedings of the 8th ACM workshop on Privacy in the electronic society (pp. 31-40). ACM.

[19] Murdoch, S. J., & Danezis, G. (2005, May). Low-cost traffic analysis of Tor. In Security and Privacy, 2005 IEEE Symposium on (pp. 183-195). IEEE.

[20] New Tor Denial of Service Attacks and Defenses. (2014, January 24). Retrieved March 23, 2016, from https://blog.torproject.org/blog/new-tor-denial-service-attacks-and-defenses

[21] O'Gorman, G., & Blott, S. (2007). Large scale simulation of tor. In Advances in Computer Science–ASIAN 2007. Computer and Network Security (pp. 48-54). Springer Berlin Heidelberg.

[22] Onion routing. Retrieved March 23, 2016, from http://www.onion-router.net/

[23] OSI model. (n.d.). Retrieved March 23, 2016, from https://en.wikipedia.org/wiki/OSI_model

[24] Pappas, V., Athanasopoulos, E., Ioannidis, S., & Markatos, E. P. (2008). Compromising anonymity using packet spinning. In Information Security (pp. 161-174). Springer Berlin Heidelberg.

[25] 'Peeling back the layers of Tor with EgotisticalGiraffe' – read the document. (2013, October 04). Retrieved March 25, 2016, from http://www.theguardian.com/world/interactive/2013/oct/04/egotistical-giraffe-nsa-tor-document

[26] Salo, J. (2010). Recent Attacks On Tor. Aalto University.

[27] Schneier, B. (2013, October 04). Attacking Tor: How the NSA targets users' online anonymity. Retrieved March 23, 2016, from http://www.theguardian.com/world/2013/oct/04/tor-attacks-nsa-users-online-anonymity

[28] Syverson, P. (2011, December). A peel of onion. In Proceedings of the 27th Annual Computer Security Applications Conference (pp. 123-137). ACM.

[29] Tor attack tries to decloak anonymous users (Wired UK). (2014, July 14). Retrieved March 13, 2016, from http://www.wired.co.uk/news/archive/2014-07/31/tor-security-decloaking-attack

[30] Tor Metrics. Retrieved March 23, 2016, from https://metrics.torproject.org/

[31] Tor Research Home. Retrieved March 23, 2016, from http://research.torproject.org/

[32] Tor security advisory: "relay early" traffic confirmation attack. (2014, July 30). Retrieved March 23, 2016, from https://blog.torproject.org/blog/tor-security-advisory-relay-early-traffic-confirmation-attack

[33] TorStatus - Tor Network Status. Retrieved March 28, 2016, from http://torstatus.blutmagie.de/

[34] Tor suffers traffic confirmation attacks. Say goodbye to anonymity on the Web. (2014, August 2). Retrieved March 13, 2016, from http://www.techtimes.com/articles/11711/20140802/tor-suffers-traffic-confirmation-attacks-say-goodbye-to-anonymity-on-the-web.htm

[35] Tor. Retrieved March 23, 2016, from http://www.torproject.org/

[36] Traffic Analysis. (n.d.). Retrieved March 23, 2016, from https://en.wikipedia.org/wiki/Traffic_analysis

[37] Traffic correlation using netflows. (2014, November 14). Retrieved March 23, 2016, from https://blog.torproject.org/blog/traffic-correlation-using-netflows

[38] Wang, X., Luo, J., Yang, M., & Ling, Z. (2011). A potential HTTP-based application-level attack against Tor. Future Generation Computer Systems,27(1), 67-77.

[39] Who uses Tor? (n.d.). Retrieved March 28, 2016, from https://www.torproject.org/about/torusers.html.en

[40] Wright, M. K., Adler, M., Levine, B. N., & Shields, C. (2008). Passive-logging attacks against anonymous communications systems. ACM Transactions on Information and System Security (TISSEC), 11(2), 3.

[41] Zhang, Y. (2009, October). Effective attacks in the tor authentication protocol. In Network and System Security, 2009. NSS'09. Third International Conference on (pp. 81-86). IEEE.