University of Amsterdam
System & Network Engineering

Security of Systems and Networks

Entry point distribution in overlay networks

Authors:
Kevin de Kok
kevin.dekok@os3.nl

Marek Kuczyński
marek.kuczynski@os3.nl

Pieter Lange
pieter.lange@os3.nl

Sebastian Carlier
sebastian.carlier@os3.nl

Abstract

Overlay networks can provide anonymity on the internet. These anonymity networks also increasingly used to circumvent local censorship by oppressive regimes and company firewalls. The combination of anonymity and censorship circumvention prove to be a powerful tool for whistle blowers, dissidents and the fight for free speech in general. As recent as spring 2010 opponents have found methods to prevent users within their realm of influence access to such a network by blocking the entry nodes on their own networks. This paper discusses several possible extra distribution vectors and a general method for personalized distribution of entry nodes, making complete enumeration of all entry nodes for a local network impossible and thus ensuring availability to the end-user.

January 3, 2011

Contents

1 Introduction

There are various overlay networks worldwide that can facilitate extra security and/or privacy to the end user. These overlay networks can be accessed without issues in most of the western world, but there are several countries that limit or block the access to such networks because of censorship.

Anonymous overlay networks are often used by people living in countries that control access to the internet. Some of the users include freedom activists, whistle-blowers, military services and people that are closely monitored by their environment (i.e. within a company). The lack of a free internet can often cause to an isolation from the truth, this is especially the case in countries like China and Iran where the access to information is fully controlled and monitored by the government.

Governments in some countries block the access to these networks by blocking the IP's that are needed to access the overlay networks. These IP's addresses are often enumerated from the networks address distribution methods, so it is of utmost importance to distribute the addresses in such a way that an adversary can only obtain as little knowledge of the network as possible while keeping the network accessible to a regular user.

One of the largest and most active overlay networks worldwide is Tor (The Onion Router) [1]. This network enables anonymous access to internet services by routing the traffic through an encrypted and untraceable path. We have chosen to focus our research on improving this anonymous service for usage in territories with restricted internet access, our motivation for this choice can be found later on in this report.

In order to use Tor, a user must access the Tor network and retrieve an entry node IP address to the Tor network. In the western world this is done through a central database, but in countries where adversaries control the internet they can block this with ease.

To get access to the Tor network in a decentralized way, an IP from a so called entry node needs to be entered by the user. There are currently no reliable and accessible distribution methods available for this, which is something that we do research in this report.

1.1 Research questions

Main research question:

"Is it possible to prevent enumeration of entry points to an overlay network?"

Sub questions:

- What are the requirements of distribution for the client? (e.g. authenticity)
- What are the attack vectors for the adversaries regarding:
  - Blocking entry points;
  - Attacking distribution points.
- Can adversaries be detected (and denied further access to the distribution system) after abusing the distribution methods?

1.2 Approach

The project approach is done with the following steps:

- Research into currently implemented methods and their failings;
- Research into acceptable solutions for a reputation system;

- Define constraints and requirements for new distribution methods;
- Propose new distribution methods that adhere to previously mentioned contraints.

2 Theoretical research

Our primary example for the distribution of IP addresses is based on onion routing. This overlay networking technique enables anonymous access to web services by routing the traffic through an untraceable and fully encrypted path across multiple nodes, meaning that an adversary cannot trace the full stream of traffic flowing between point A (the user) and B (a server). The path is routed by the client based on information given by the onion routing directory services, after which the client encrypts the data packet with layers of encryption (similar to an onion), where every hop can only decrypt one layer of protection. Examples of successful implementations of onion routing are Tor and I2P (with Tor being superior over I2P regarding exposure, finance and code development).

There are three crucial components within a properly designed overlay network;

- Entry nodes
- Relay nodes
- Exit nodes

2.1 Entry nodes

Any client that uses the Tor network needs to reach an entry node. This node then forwards the encrypted packet to the next predetermined hop within the Tor network. Keep in mind that the traffic can still be intercepted before it is encrypted by the entry node.

2.2 Relay nodes

These nodes are located in between the Entry and Exit nodes, they relay the network traffic through a random path. A relay node knows only the address of the previous and next hop of a data packet, but not the source or destination. Tor usually uses just one relay node while I2P is designed to use multiple, even though this does not necessarily increase anonymity.

2.3 Exit nodes

The traffic relayed through Tor/I2P exits the network here from where it continues to its final destination. The traffic is unencrypted between the exit node and the destination unless i.e. SSL is used. The maintainers of exit nodes must be aware that illegal traffic (i.e. child pornography or terrorism communication) might exit the Tor network through their exit node.

Traffic can safely traverse through the overlay network once it has reached an entry node. When adversaries do not allow the use of Tor in their country often block access to these entry nodes and to the distribution methods used to retrieve these addresses. It is know that China has been blocking access to both over the last few years[9], but the same has been happening in Iran as well[10] . The Tor project group has tried to come up with solutions in order to prevent enumeration of all nodes and to keep the distribution methods accessible, but the current methods were not built with this in mind.

Bridge relays were introduced into Tor, which are equal to regular entry nodes, except that their addresses are not listed in a public directory. Instead, the Tor project maintains a separate database containing these addresses (called bridgedb), which are then distributed in small batches through (currently three) different mechanisms. Enumerating these addresses is a lot harder and more time-consuming for an adversary because only the bridgedb maintains a complete overview of the bridge nodes.

I2P is affected differently in restricted area's, since its address allocation is a lot more distributed; it uses decentralized "eepsites" to announce node addresses within the network. These do not contain a full listing of all nodes, but just a small slice to get the client connected to some peers.

2.4 Proxy servers

Proxy servers are one-hop relays that can be used to circumvent a restricted internet zone. They do this by relaying all traffic through a single point located in a place where there are no restrictions on internet usage. The advantage of this method is that it is not hard to configure and easy to use in different application layer protocols. A big disadvantage is that you need to trust the maintainer of the proxy and the party's that are between your connection and the proxy server itself.

2.5 Freenet

Freenet is an example of a decentralized P2P overlay network where content (message boards, files, media) is stored across multiple random nodes in encrypted blocks. Another term used to describe this type of network is internal caching network. It's not really comparable to i.e. onion routing since it does not act as a proxy but like a "small world network" where people can host their content in a distributed way. The owner of the content cannot be traced back, and even the current location of the distributed content cannot be found because of the encryption.

Nodes that participate in Freenet need to use a special client which tries to find neighbours to connect to. Whenever a request for a file is made, the client asks its neighbours (even over multiple hops!) until the content is found. This works up until a certain level, after which content simply cannot be retrieved. Alternatives to freenet include GNUnet, Opennet, Darknet and Entropy, they all work in a similar fashion.

3 Current methods

In this chapter, we will look at the various strategies that are being used to distribute entry-nodes in overlay networks. In order to investigate this we looked at how the current overlay networks distribute entry node addresses in territories that have adversaries controlling the internet.

The current methods used by Tor to circumvent censorship are listed below;

- HTTP bridge distribution
- Email bridge distribution
- Manual bridge distribution

In all methods trustworthiness is an important vector. Also scalability of the distributions is very important. Not all methods are providing those vectors.

3.1 HTTP bridge distribution

The Tor project currently has a website available where bridge IP's can be retrieved, see https://bridges.torproject.org/ Every time the web page gets accessed, the client is presented with three bridge IP's which can be entered manually into a Tor client in order to connect to the network. An adversary that has a lot of IP addresses available can still enumerate a large share of the entry nodes. It is also very easy for an adversary to block access to this web page (either by blocking keywords in the URL or the IP address it's currently hosted at, this is currently the case in China).

Both the HTTP and E-mail distribution methods use a consistent hashing algorithm to maintain a list of what client has received what information. This basically boils down to making hashes of all the bridge IP addresses listed in the bridgedb and hashing the requesters IP's. Then the two hashes are compared, and the requester will receive three IP addresses that have very similar hashes to the hash calculated for him/her by the server.

3.2 Email bridge distribution

The Tor project also uses address distribution through the Gmail and Yahoo web-email services. This service works only with these two mail-services since they offer some protection against mass e-mail account creation. In order to get bridge addresses, users have to send an e-mail to bridges@torproject.org, after which they will receive a reply containing three bridge IP's. The e-mail distribution uses a similar anti-enumeration to the one provided by the HTTP bridge distribution. Gmail provides another feature called dkim[11]. This feature signs every outgoing email. If it didn't an attacker could spoof addresses from Gmail and spam them. After a while, Gmail could tag us as a potential spammer and none of those mails would get through any more.

3.3 Manual bridge distribution

Bridge addresses can be entered manually into a Tor user client. These can be obtained in various ways, as an example through an university or organisation or through an internet relay chat (IRC) channel. The big advantage of this is that the addresses can be kept "private" within a small group, but the disadvantage is that it does not scale at all and that you need some sort of web of trust. You'll also need someone who does have access to a part of the bridgedb or one of the conventional methods, which can be hard in restricted area's.

4 Proposed transport and storage mechanisms

In this section the developed transport and storage mechanisms and the are discussed. The distribution methods are used to distribute entry node information amongst users and the storage mechanisms are used to store the payload in a safe manner. The next chapter describes the the practical implementations of the components described below.

4.1 Transportation Methods

The transportation nodes are based on different services which do not depend on each other. Those mechanisms are used to transport information about the entry nodes to the end user.

4.1.1 IRC

Internet Relay Chat(IRC)[4] is a client/server technique that can be used for chatting with multiple users. A client needs an IRC client to connect to a IRC server, after which it can join a chat room to talk with other users. Multiple IRC servers can form an IRC network, which means that the network can have multiple entry points. Some IRC networks provide IPv6 and even SSL connectivity, meaning that the communication between the client and server is encrypted.

IRC channels consist of multiple users and possibly a channel bot. A bot is a non-human client that can react on predefined input from a human. The basic operation of a bot is giving users operator status in a channel so that users can manage the channel and can remove people who are misbehaving, but they can be extended to automate other tasks (like handing out bridge IP's) as well.

Even though the IRC traffic can be encrypted end-to-end, an adversary can still see who is connecting to an IRC server, after which the IP traffic to the IRC server could be blocked. This can be circumvented by the use of a BNC, which is software to create a proxy for IRC. This mechanism provides some sort of privacy on an IRC network because there is a hop in between, meaning that the connection is harder to trace and that users on the IRC network cannot see the clients IP address he uses at home. File transfers are also possible between the client and an IRC network through a bouncer.

Other ways to connect to IRC is by using a so called jump host. A jump host is a shell server which is accessible on the internet through which you can connect to an IRC server using an IRC client on the server, for example by using the Irssi program on a remote Linux server. This is different from a BNC, which acts as an intermediate IRC server the client connects to directly.

4.1.2 Freenet

Freenet is an open source project that focusses on content storage on a very distributed content overlay network. Clients that participate in Freenet have to install a special client which downloads a part of the network's content after it has set up an encrypted connection to other nodes. The content on Freenet is stored in encrypted chunks that are spread out over multiple Freenet nodes. If content is requested within network, broadcasts are send out until the piece of data can be located at a peer or a few hops further, after which it gets transferred to the requester.

Freenet can operate in two different modes to retrieve nodes to connect to; Opennet and Darknet.

Opennet relies on seednodes to retrieve a partial view of the Freenet network. These seednodes can be defined manually in a configuration file or the default seednodes can be used (this is also called the bootstrap). After a seednode is contacted, the Freenet node will receive a partial view of the Freenet network, after which it can operate on the network. This method scales fairly well because the client only needs the bootstrap to get connectivity to the network. Manual seed nodes can be entered to omit reliability on blocked seednodes, this can be useful in countries where adversaries have control over the internet, although it obviously scales less well.

Darknet relies on manually configured Freenet nodes that the user must enter in order to access a private Freenet network. While this does not scale at all, it does provide a very good security

model as long as the admission to the network is verified. The network is not advertised in any directory and it is pretty hard for an adversary to detect that there actually is a Freenet network operating.

4.1.3 BitTorrent

BitTorrent can distribute files without having a heavy load on the network or on client machines. The principle behind it is by involving users into the distribution of files; users join a swarm of hosts while downloading and uploading directly from each other. This mechanism provides also the ability to distribute files from clients which have low bandwidth. When a user wants to upload a file he creates a torrent descriptor. This file should be uploaded through a website for example. The user is going to participate in the network as a seed which means that the user is offering the file to other torrent nodes. A file which contains the IP addresses of the bridge servers can be distributed over the BitTorrent network. Another technique which does not depend on BitTorrent can be used to distribute the passphrase that is needed to read the file.

4.1.4 Skype

The Skype overlay network can be used in various ways to distribute entry node IP's, this could be done through voice calls, chat messages, group messages or file transfer. Skype conversations between two clients are encrypted by 256 bit AES[12] session key[13] and most of the protocol is based on common standards. However, not all of the protocols used have been disclosed to the general audience, so it is not known if there aren't any actual backdoors present in the software. The Skype network is fully accessible from China and it is even operational through firewalled and NAT-ted environments. Skype also provides an API that can be used to automate conversations and therefore create a bot that responds to clients, which does make it a very accessible and elegant transport mechanism.

4.1.5 Gmail

The current Gmail implementation is fully SSL encrypted, which makes it a suitable but very accessible transport method. An adversary might have access to the traffic information or the actual e-mail content (i.e. Google has to comply with non publicly disclosed regulations in China), so it would be advisable to perhaps encrypt the payloads. It's still a very easy and accessible method to use.

4.2 Storage mechanism

To transport the information without getting eavesdropped by an adversary encryption methods are needed. Different available encryption mechanisms can be used to encrypt the data.

4.2.1 Symmetric encrypted file distribution

Symmetric encrypted files can be decoded if a user knows the password or has the key file. They offer a very scalable solution since the end-user only needs the crypto file and the password or key file, but it is also very hard to distribute this safely to a large group of unverified users - it takes just one rotten apple to enumerate the contents of the file to an adversary. This risk can be mitigated by making the user groups smaller, but the solution obviously becomes less optimal then.

4.2.2 Asymmetric encrypted file distribution

Asymmetric key encryption is based on a public and private key which every user holds. The private key of a user is used to sign or decrypt data while his or her public key can be used by a third party to verify the authenticity or to encrypt a message. This is not scalable in the sense

that there needs to be a bond between two parties, but this can be solved by using a public key infrastructure (known as PKI).

4.2.3 Gnu Privacy Guard(GPG)

GPG [5] is a implementation of OpenPGP[6], this mechanism can be used to encrypt and decrypt files. The encryption mechanism that is used with GPG is called public and private key encryption. The users who want to use GPG create public and private key pairs on their PC which is tied to an email address and a user name. The public key of a user can be uploaded to a so called key server, after which other users or applications can retrieve the public key.

4.2.4 Plain text

Plain text communication might sound like a very obvious option, but encryption does not have to be a necessity if the transport mechanism provides enough security or when the plain text information is not comprehendable for the adversary without additional information.

5 Reputation system

5.1 The goal

The main cause for designing a reputation system is the limited availability of resources, namely addresses of entry nodes. Because we want to enable everyone access to the network these scarce resources are currently released in limited amounts to anyone who requests for them.

The important and sometimes conflicting aspects of this method of distribution are:

- Anonymity of the user
- Quick/easy availability to the user
- Limited amount of resources to spare

The anonymity of the user and the availability of working entry nodes to that user are fixed design parameters of the system. The limited amount of resources is a result of the nature of most anonymity networks; they operate using resources provided by volunteers. If we find methods to reduce the *scarcity* or *value* of the entry nodes as a design parameter we enable ourselves to build a system that can simply dispose of failed (blocked) entry nodes. What defines a resource as valuable is its IP address; if the IP address is dynamically assigned to an entry node then blocking that IP by some type of authority is not a terminal threat to the system. The machine can for example be automatically rebooted to obtain a new address. Below are the requirements of a machine that is considered expendable to the system:

- easy/fast to start
- scalable in terms of amount of machines that can be started
- receiving an IP address from a large pool after a reboot
- provided with sufficient bandwidth

The listed requirements match machines that can be spawned in a cloud environment.

This solution in essence splits up the resources into two groups; the expendable entry nodes and the static entry nodes. The disadvantages and advantages of both groups are listed below.

	cloud node	static node
disposable	yes	no
cost	yes	no
limited	by cost	by volunteer participation

Thus, our ultimate goal is to enable users to obtain access to the overlay network even while an adversary is actively trying to enumerate available entry nodes and locally block access to them.

5.2 Reputation systems basics

Reputation or 'karma' systems usually serve the purpose to judge and evaluate peers of a given system. It is done so that peers can decide wether to trust others with either transactions or information. The types of karma systems are:

- Participation karma - users get graded based on their involvement in the system. They are graded by an algorithmn. (eg. calculating number of posts on user forums)
- Quality karma - users get graded based on the quality of interaction with others, not on the frequency of interaction. They are graded by their peers. (eg. Ebay)
- Robust karma is a combination of the above.

Reputation foundation for our system A quality karma system can not be considered a solution because there is minimal involvement from the user. The user does not interact with his peers within the boundaries of the system. On the other hand participation karma does not require the involvement of peers. It can be calculated by an algoritmn, and can be done with some level of anonymity. Because the user cannot contribute anything to the system, the participation has to be calculated not from the users non-existant contributions, but from how long he has been in the system. Obtaining sufficient reputation enables the user to obtain stable resources.

Calculating reputation It is only important for the system if a user is involved in blocking the entry nodes he obtained. If that does not happen during ones trial period then that user becomes trusted.

The trial period needs to be sufficiently long and should be calculated during testing. To be able to identify which user leaked the resource, the system gives out distinguishable resources to different users. The user receives three entry node addresses based on the first X bits of his IP address and based on the time he requests the entry nodes. The usage of the user's network address prevents easy enumeration of entry nodes by a potential adversary that owns many hosts in different subnets; it is already implemented in Tor.

At specific intervals the list of entry nodes is reshuffled and new sets of three different IPs are formed. This enables us to keep a sufficient trial period, yet keep the numbers of users that received the same resource at a comfortable limit. The interval should be calculated during testing. It is a compromise between false positives and striving for anonymity.

Requesting a new trial period is possible, so resetting every suspect's reputation is considered acceptable. We assume that an adversary is not easily identifiable therefore it is pointless to pinpoint him or blacklist suspects.

- The system itself does not require registration, just the knowledge of the resource
- The user is not being identified by a login system
- The user is not uniquely identifiable by the resources he uses.

The users identity can only be verified by a pre-shared key upon his request. This occurs when a user requests stable resources and is aware that he obtained enough reputation before the request. As stated at the beginning of this chapter one of the requirements of the system is anonymity of the user. Therefore the solution does not provide any type of user registration.

If a user leaks a distinguishable resource, all of the users that received that same resource (and are still in the trial period) loose their reputation. As the entry nodes designated for that subnet are blocked, new entry nodes are spawned and divided into groups of three. Those triplets are assigned only to smaller blocks of that subnet and must not be given to any other subnets. As of now those blocks are unique IP addresses. Users that come from those IP spaces (and request new entry nodes) need to be identified with a cookie they previously committed to the system. This is done for the following purposes:

- further dividing the subnet puts the adversary in a smaller group of users. Thereby effectively giving the group that does not contain an adversary a chance to build up trust and lets them use the new nodes.
- if further splitting is not implemented an adversary can request new entry nodes each time he blocks the ones he obtained, repeating the process indefinitely. In practice preventing building up trust to other users from his subnet and making the nodes obsolete.
- identification by cookie needs to be done upon the second request, so that the adversary cannot spawn a large number of hosts in smaller address spaces blocking even more resources than during the first step.

It is still to be determined how many times a subnet should be split up in case of an incident and into how many subnets it should be split up. Although during each incident a users reputation should be reset to 0 or severly lowered. As mentioned previously as of now the split only occurs once and each of the users is given a unique set of IPs after a blocking event occurs.

To keep anonymity, users do not log into the system, they are identified by a hash of a cookie with salt. This only allows us to keep two states in the karma system. A new state and a trusted state the user can obtain. There is no point to blacklisting users that try to harm the system by their cookie, because it is too easy for a perpetrator to change his identification and start harming the system again. Blacklisting by origin (IP address) is still an option.

5.3 Algorithm

The algorithmn for building trust of new users consists of two main events. Those events are:

- Request for nodes
- Blocked node reported

Apart from the above events there is a periodical reshuffle of the database holding the entry nodes. This reshuffling increases the efficiency of the algorithm. All of the three parts forming the whole algorithm is described in the sections below. They are described in detail with the reasoning behind the choices made.

The algorithm also uses four different tables that are described as databases in this chapter. Those are:

- trial_bridgedb ; keeps the data about entrynodes and subnets they were assigned to.
- trail_userdb ; keeps entries describing each user that is on trial.
- div_userdb ; keeps entries describing each user that received its own uniqe set of IPs of entry nodes.
- bl_userdb ; keeps IP addresses and cookies of users that are trying to attack the system.

5.3.1 Trigger Event: Request for Nodes

When a user requests nodes the algorithm described below is follwed:

- The source IP is checked against a database cointaing blacklisted addresses, called bl_userdb. If it is found the request is terminated. This prevents an attacker from comprimising the nodes from the same address attacking with a time interval.
- The source network of the attacker is checked to see if entry nodes have alredy been assigned to that subnet. This is checked in a database containing the entry nodes, called trial_bridgedb. If the source network is not found in that database it is equivalent to no connections being made from that network previously and no nodes allocated to that network. Therefore the following actions occur:
  - The source network is added to the database with three new entry nodes allocated to it.
  - A new entry in trial_userdb (a database containing user details) is created. The entry stores: the source IP, the entry node addresses that are sent to the user, a newly generated unique cookie for the user and the time of the event are stored. The cookie is an identifier for the user and the timestamp determines when the user can be trusted.
  - the entry node addresses and the cookie are sent to the user.
- If the source network is alredy in the database, that means users from that network already requested entry nodes and that there are entry nodes allocated for that network. If the user does not identify himself with a cookie, a new entry in trial_userdb is created for that user and the node addresses for his source network are sent to him. The entry is created in the same way as described above. Also the user receives the entry node addresses and the cookie.
- If the user identifies himself with a cookie and that cookie does not exist in trial_userdb, the user is restricted from requests for a short amount of time. His source IP is put on a blacklist. This is done to prevent an adversary from a bruteforce attack by guessing cookies, trying to impersonate a valid user.

- If the user identifies himself with a cookie and that cookie exists in trial_userdb and enough time has passed since the user's first request the user obtains the stable Tor entry nodes. They are not part of this protocol.
- If the user has not been in the system for long enough he receives the node addresses from his entry in trial_userdb.
- If a blocking of the nodes (allocated to the user's network) occured, the users penalty variable is increased by one. In that case the user receives three entry nodes from a different set.
- The algorithm checks if the user is requesting nodes for the first time after the blocking event. In that case a unique set of nodes is assigned to that user. The source IP, cookie, time of the event and the new entry nodes are stored in div_userdb. Giving unique sets to different users from the same subnetwork makes it harder for an attacker to spoil everyone's trust from that subnetwork. That way even with multiple attackers in a subnetwork, users can still build up trust. The algorithm spawns a minimum amount of bridges for that occasion, the uniqness of sets is cheaply achieved. The number of users with unique sets equals $\binom{n}{3}$, where n is the number of spawned bridges.

5.3.2 Trigger Event: Blocked Node Reported

Another important event is when nodes are blocked, this event determines that at least one user has broken the trust. Considering an attacker has limitless resources it would be futile to try and block the attacker from using Tor based on his IP address for example. We assume that an attacker would like to block all of the entry node addresses he obtained, as blocking only one or two does not stop other users from connecting to Tor. Therefore the following algorithm is taken in case of a blocking event:

- The node is marked as blocked in trial_bridgedb and the time the node is reported blocked is saved. The machine that was running the node is also shutdown and reset during the next reshuffling.
- The subnets the node belongs to and belonged to are are checked. If the other two nodes from one of those subnets are also blocked around the same time that means that a user from that subnet is an attacker.
- User's from that subnet recieve a penalty point and the next time they request new nodes each one of them will be given a unique set of entry nodes. This should happen soon as users will not be able to connect to their old nodes (as they are blocked).

5.3.3 Time Triggered Reshuffling of trial bridge database

The trial bridge database is reshuffled periodically. This period should be at least half the time it takes for a user to become trusted. It is done so that a set of bridges given to a group of users is not only unique for a certain subnetwork, but also unique for the period the users first requested the nodes. This has a purpose of not creating groups that become too large as the risk of getting an attacker in a group grows with each user joining. If the group is big and a blocking event occurs too many users need to receive unique sets and to many users breach the system's trust. For the reshuffling the following algorithm is followed, when enough time has expired since the last reshuffling:

- The variable indicating the last reshuffling is updated.
- Every third entry counting from the first one, should not be permuted as they contain only the subnet address.
- Every third entry (starting from the second one) is permuted according to the following algorithm (algo from wiki, sorry dont have the link atm)

- The same permutation is repeated on every third entry starting from the third one and from the fourth one.

5.4 Attack scenarios

This section lists certain attack scenarios an adversary might use to block our system. It also explaines what measures were taken to prevent those attacks from happening or to severely limit them.

Floooding entry nodes with traffic To prevent an adversary from utilizing all of the bandwidth provided for the nodes, therefore increasing the cost of upkeeping the nodes, each node needs specific firewall restrictions. A node that is ment for specific subnets, should only allow traffic from those subnets. Therefore crippling an attackers attempt to flood any given node from any location.

Enumerating every dynamic node Enumeration of every node is prevented by supplying different addresses to different subnets. In essence an attacker can only get more nodes if he connects from different subnets. Therefore the subnets need to be sufficiently big so that one can not easily enumerate too much of the network. Moreover this approach becomes ineffective, because the nodes are restarted with a new address. Even though partial enumeration can not be prevented, the damage can be repaired by restarting blocked or spawning new nodes.

Blocking all of the possible cloud service addresses Although ths is a valid atack scenario, it is assumed that the perpetrator will not take such great measures. Cloud services are becomieng an increasingly important part of the Internet. If the whole address space of a cloud provider where to be blocked, it would rid certain businesses or research facilities of on demand computing power.

Blocking a specific subnet from a stable connection to an entry node This attack scenario is partially solved by implementing a tier solution for giving out entry nodes; by splitting a suspected group into smaller groups on the base of supplied entry nodes.

6 Analysing the components

In this section the important requirements are written about the distribution and the encryption methods. All methods are compared by using the following requirements:

- Scalability
- Encryption
- User friendly
- Anonymity
- Single point of failure

6.1 Transportation Methods

In this section the transportation methods are discussed.

6.1.1 IRC

Scalability IRC scales very well on the internet. The current known available networks are distributed over a lot of servers on by different isp's all over the world. One must keep in mind that everyone can begin his own IRC network and provide services over the IRC protocol to users.

Encryption IRC provides the ability to use SSL. By using SSL it is impossible to eavesdrop on the connection and intercept the plain text that is communicated between the client and an IRC server. The so called bouncers also provide the ability to use SSL.

User friendly Connection to an IRC network can be done easily by a user. There are several IRC clients that can be used on different operating systems to connect to IRC networks. The configuration part on a client is adding a IRC network and connection to that network and joining a so called chat room.

Anonymity Anonymity is guaranteed on IRC networks. Different networks provide a service called cloaking. By cloaking the actual host from the client is hidden for the entire network. The user can also use a bouncer to connect to the IRC network. The user connects first to the bouncer and the bouncer connects to the actual IRC network. The originating host from the user remains hidden on the network. Only the hostname from the machine which hosts the bouncer is known.

Single point of failure The single point of failure in IRC networks depends on the fact that only one server is available for a network. Most of the known IRC networks are connected through IPv4 and IPv6 by different network providers, thus providing entry points to the irc network on different IP addresses.

6.1.2 Freenet

Scalability Freenet is very scalable because it can retrieve information (content and Freenet nodes) from the network once it has been bootstrapped. Every node maintains a partial view of peers within the network and it can search through the network by contacting them.

Encryption Freenet is fully peer to peer encrypted and it is close to impossible to detect it is being used when the user chooses Darknet mode.

User friendly In our experience it was very easy to set up Freenet on an Ubuntu system. The user needs to install a piece of software after which the network bootstraps information from static nodes defined in the configuration. It will take a couple of days before (a part of the) content is replicated onto the user's PC and for the network to get up to speed, but after that it will work just fine for the purpose we need. Even a poorly experienced user could probably get this running.

Anonymity Increased anonymity is basically why Freenet was created for, but it can be applied in very different levels in the program. While it is possible to see what peers you are connected with (in both Opennet and Darknet modes), it is not possible to see the origin of the content or where it is currently stored.

Single point of failure None really, unless the bootstrap nodes get blocked. This can be overridden manually though after which the Freenet network should work just fine. A very strong advantage of Freenet is that the content is replicated over multiple locations, meaning that it is hard to trace back the source or to completely delete content from the network.

6.1.3 BitTorrent

Scalability BitTorrent scales very well on the internet. Users theirself who download a specific torrent file contribute imediatly on with distributing the file itself. They will are added in a so called swarm. By this mechanism every single user contributes with uploading a piece of the content. Once a torrent file is distributed it is hard to get it offline again.

Encryption The files that are downloaded over torrent are plain text. There is no encryption by using the default available BitTorrent clients. However, there is a solution available, this solution is called TorrentPrivacy. This client offers encryption by tunneling over the SSH protocol between clients. This makes eavesdropping in the sense of reading the actual data impossible.

User friendly The BitTorrent protocol provides clients for multiple platforms. It is basically a normale program with an installer. No advanced computer knowledge is needed to configure the software.

Anonymity BitTorrent itself does not provide any form of privacy. The source ip's from the participants are known in the tracker. However, TorrentPrivacy offers solutions to that. This problem is solved by using a webproxy. The originating ip from the client remains unknown. Only the ip of the web proxy is known at the tracker.

Single point of failure The torrent files are distributed over several trackers. Those trackers can be public or private. Since there are multiple trackers, single point of failure is eliminated in the sense of obtaining torrent files. By the nature of the BitTorrent protocol it is impossible to have a single point of failure. Because file sharing with this protocol is distributed over the participants who are interested in one particular file.

6.1.4 Skype

Scalability Skype us just one basic program that needs to be installed. An Skype account is required to make usage of this service. This service is available in every country on the world. However, in some country's a user may be offered a different version of Skype. The default client also works in those country's.

Encryption Skype makes use of the current known used encryption algorithms, like AES and RSA. The conversations between clients are fully encrypted. One can argue if privacy is fully guaranteed, Skype does not confirm if they operate with governments regarding to eavesdropping on Skype conversations[18].

User friendly Skype is very user friendly in the sense of installing and using the service. The installer is easy to use and the ways to manage the contacts and other important settings are well documented. It is also very easy to set up the accounts.

Anonymity Skype lacks the feature of being anonymous, there are possibilities to identify Skype traffic, even over overlay networks. One must take into account that this is very hard. And even harder when doing to this on large scale with a lot of Skype traffic. Skype can also use a proxy server to connect to the Skype network. The originating IP from the client remains unrevealed by using this proxy.

Single point of failure The single point of failure remains in the login servers from Skype. If those servers are blocked, then it is not possible to make usages of Skype anymore.

6.1.5 Gmail

Scalability Gmail service is available in every country. Since this is a web based service, no additional software is needed on the clients. A client connects with his web browser to Gmail and uses this service in his browser.

Encryption Gmail makes usage of HTTPS to login to the web page. By using Gmail-to-Gmail communication no information is send out of the Google network in plain text. So eavesdropping is not possible on the Gmail email infrastructure. Gmail also provides dkim support. This feature ensures that originating domains cant be spoofed.

User friendly Gmail is user friendly and easy to use. A user surfs with his web browser to the Gmail website and use his Gmail account to login to the website.

Anonymity There is no anonymity with Gmail. Every eavesdropper can conclude that a user is connected to Gmail by analysing the destination ip. But, since this is a email service and the connection is encrypted between the client and server, some form of anonymity is guaranteed.

Single point of failure The single point of failure relies in the fact that Gmail can be entirely blocked. But, by using a proxy server this problem can be solved.

6.2 Encryption Methods

In this section the ways to encrypted the data is being discussed. Different methods of encryption are being used.

6.2.1 Symmetric encrypted file distribution

Symmetric encryption can for example be used when releasing a encrypted file over BitTorrent. The benefit is that only one static key is required to decrypt a file. The key can be distributed by using a possible applicable transportation method. The key is not bound to a personal identity compared with asymmetric encryption. It can only be used when distributing entry nodes with a transport mechanism that reaches more then one user.

6.2.2 Asymmetric encrypted file distribution

Asymmetric encryption can be used when an user requests a list of entry nodes that should point to a personal identity. The personal identity is the combination of his Gmail address and his gpg key. This information is not shared with other users. The public key is used to encrypt the data and the user can decrypt the data with his private key. Gpg can be used for asymmetric

encryption, this form in implementation of asymmetric encryption is used in different parts in the project.

6.2.3 Plain text

Plain text can be used when a overlaying form of encryption is used. If a transport method provides that encryption layer then plain text can be used. Since the encryption is provided by the transport methods, eavesdroppers cant read the plain text.

7 Combining the proposed methods

The following options could be combined in order to create entry node distribution methods. Some of the methods presented are very trivial, but they might provide a solid solution when they are combined together. Two of the propositions are worked out as proofs of concept in the next chapter.

7.1 Freenet (darknet)

The Darknet mode of Freenet does not scale very well, but it could be used to distribute entry nodes within a small group of mutually trusted users (i.e. students at a university). The package itself contains all the tools needed for secure communication between known peers, but it does not directly solve the problem of acquiring entry node IP addresses, it just makes the distribution of working addresses easier between users.

7.2 Freenet (opennet) & GPG

Everyone can access Opennet, which means that an additional security is needed if this is to be used for bridge distribution. GPG could be used for the encryption of the payload, after which the end-user can decrypt the message on his/her PC. This process can even be automated, although it does take some time before the file propagates on a larger opennet.

7.3 IRC & GPG

IRC can be used to obtain access to a distribution point of entry nodes. The distribution point is an IRCbot where requests can be made by a user on a IRC network. IRC provides the interface to the end user. The service, the so called IRCbot is the communication point of the user. The user must provide a Gmail address and his gpg id. The IRCbot will verify if the email address corresponds with the email address provided with the gpg id and encrypt the list with entry nodes and email it to the user.

7.4 Skype & plain

Skype claims to offer end-to-end encryption using AES256 [13], so it should be secure to transfer text or files from end-to-end directly. Skype does not give the hard guarantee that the content cannot be eavesdropped upon. It's easy to reach from around the world since it's a very distributed overlay network that is accessible even from territories controlled by adversaries, even though this might be changing in the near future.

7.5 Skype & GPG [added security]

Gpg can be added to the Skype alternative above to identify the user by his or her key. It can also offer an additional layer of security in case the adversary can snoop upon the AES-256 encrypted content.

7.6 Gmail & Plain

Gmail offers full HTTPS encryption over all the content within the Gmail website. We can therefore assume that it is reasonably safe against an adversary sniffing the content on the wire. This method is currently operational already by the use of the bridgedb.

7.7 Gmail & GPG [added security]

The same story as for the Skype + GPG option applies here - the user can be identified and an eavesdropped cannot decrypt the content if the contents of an e-mail attachment would be intercepted.

8 Proof Of Concepts

The proofs of concept (PoC's) implemented by the project group are described in this section. The PoC's can be distinguised into three categories, namely:

- Distribution methods;
- Reputation system;
- Block detection mechanism.

8.1 Distribution methods

In this section two PoC's of distribution methods is discussed.

8.1.1 IRC combined with gpg

The first distribution method is by using IRC to distribute an list with entry nodes to the Tor network. If an user wants to obtain a list of ip's via IRC he/she must have the following information and configured technical solutions:

- GPG keys;
- Gmail account;
- IRC client;
- IRC channel with request bot;
- Possible layer of anonymity and security.

To obtain the list of ip's the following procedure should be taken by the user:

1. Connect to IRC network
2. Join IRC channel
3. Query the channel bot with GPG id and Gmail address

The channel bot has its backend configured with the script provided. This script fetches the public key from the user based on its GPG id at a key server. The provided Gmail address on IRC is matched against the Gmail address in the public key information. If the Gmail addresses are not the same, the request gets discarded. If they are the same then the request if processed. Processing of the information is done by the following steps;

1. Receive key;
2. Encrypt IP list;
3. Write email;
4. Send email with attachment;
5. Remove temporary files.

The front end of the bot has a trigger configured to match the provided query from the user.

```
/trigger add -privmsgs -regexp '^0B447282 kdekok@gmail.com' -command 'exec ./gpgquery.sh 0B447282
```

The regular expression should be matched, if this is matched then the backend script to process the request is executed.

The backend script matches the information provided on IRC with the information in the gpg public key. If that information correlates then the request is processed and the entry nodes are send to the corresponding email address. The backend script can be found in appendix A.

8.1.2 Skype with GPG

The second developed proof of concept is based on a Skype bot which will supply the user with bridge addresses on request. It can utilize the same gpg authentication method described in the paragraph above, but it retrieves and processes this information in a slightly different way. The project group was able to build a simple auto-replying bot using the Skype API and has also created a simple Python script which a client can execute to contact the Skype-bot directly. This can also be done from the Skype client itself, but the user would have to send the bot a friend-request which has to be accepted by the server first - this would be an unwanted and non scalable extra step.

A user needs the following to retrieve bridge addresses from the bridgedb using Skype;

- GPG keys;
- Gmail account;
- Skype account (free);
- Optionally Python for the client script;

The following procedure must be followed to retrieve entry node IP's:

1. Run the Python script with the gpg key and Gmail address as arguments.

As an example of a request;

```
$ req_bridge.py gpg [choose gpg option] 1234567 [GPG ID] marek.kuczynski@gmail.com [e-mail addres
```

The program will contact the Skype API, send the request to the Skype bot and e-mail the user the gpg encrypted list of files. The Skype server runs a similar setup, namely an installation with the Skype client, Python and the code you can find in the appendix. It's configured to accept conversations from any registered Skype user, after which it will pass the arguments given in the chat to the gpg script.

Alternatively, in the event that e-mail is no longer a safe option, the response can be replied using the Skype chat window as well, but it is impossible to assign any reputation to the user when using this option. It is, however, a good backup mechanism in case Gmail does get blocked/filtered by an adversary.

8.2 Verify blocked node

The detection mechanism to verify if a Tor entry node is blocked is done by sending an TCP packet to a website hosted within a territory where outbound connections are monitored by an adversary. Testing the connection is trivial to determine whether the address has been added to a firewall blacklist. This is done by the following approach;

1. Send ip packet to website with src.port <=> tor.dst.port;
2. If SYN/ACK back; then entry node not blocked;
3. If no response; then entry node is blocked.

The lines of code for this particular PoC can be found in the appendix A.

9 Conclusion

Referring back to the research question:

"Is it possible to prevent enumeration of entry points to an overlay network?"

By answering this question the research project was split up into two categories. The first category describes the different distribution methods. Which is a combination of transport and storage mechanism. And the second category is a reputation system for distributing ip's amongst the users of a overlay network. The biggest problem to overcome is the issue of bootstrapping the end user; where is the starting point for obtaining IP addresses from bridge servers. According to Tor – our main example of an anonymity based overlay network – the people in countries where the Tor service is blocked are smart enough to distribute information about the bootstrap method amongst themselves. The transport mechanism that are described in this document are known services, that is described in section 4. Those services are chosen because they don't depend on each other. This makes the distribution better protected for getting blocked by adversaries. The information that is transported is protected by our so called storage mechanism, this is described in section 4.2. This mechanism encrypts the data so that eavesdroppers can not read the information. Two techniques are used, symmetric encryption and asymmetric encryption. The encryption techniques are distinct based on global or personal distribution of the bridge information.

Our designed reputation system fixes the current 'hidden entry node' distribution by keeping track of the trustworthiness of our end-users. Instead of distributing the same set of entry nodes to a C-class network, users will get distinct sets of entry nodes when blocking events have been detected. This enables us to reward good users who have never been associated with sets that have been blocked by adversaries, but also requires us to keep track of end-user source IP's and associated anonymity cookies'. We are unsure if it would be sufficient to only keep track of the network address or if it is sufficient to only store the IP address of users who are already in a group with an adversary. Due to time constraints it was impossible to create a working proof of concept, but many new questions have been raised. Refer to the Future Work section for more information.

10 Future work

To be able to do any further research on the subject, the proposed methods need to be tested in a real life environment. The first action to take is to try and establish some agreement with the TOR development team so that the described methods can be reviewed and implemented for a testing group.

The step after that is to calculate the number of unique users and the consumed bandwidth. This is very important because it will establish the estimated cost of the entry nodes. The costs then should be presented to the TOR development team and another agreement should be made to look for sponsorship. We propose organizations that have interest in providing channels of communication from heavily moderated states.

The technical part of the future work would be to track if distribution methods are effectively used by clients. This could be done by statistically checking how much is given out via which channels. Furthermore the algorithm that lets users build trust, should be tested to check its effectiveness, as there might be room for improvement by tweaking certain variables. This might include testing the time it should take for a user to become trusted.

A Proof of Concept code

Skype Server

```
1  ######################################################################
2  # marek.kuczynski@os3.nl, december 2010
3  # SSN overlay network project
4  #
5  # This script uses the Skype API to reply bridge addresses to
6  # a user if the correct passphrase is given. It can be extended
7  # in order to retrieve the bridge IP's from a database and to
8  # retrieve the passphrases from another source.
9  ######################################################################
10 import sys
11 import re
12 import os
13 import Skype4Py
14
15 # Check if the Skype client is running and if API communication is possible
16 def OnAttach(status):
17     print 'API attachment status: ' + skype.Convert.AttachmentStatusToText(status)
18     if status == Skype4Py.apiAttachAvailable:
19         skype.Attach()
20
21     if status == Skype4Py.apiAttachSuccess:
22        print('**************************************')
23
24 # This routine checks for inbound messages and returns bridge IP's if the correct
      passphrase is given
25 def OnMessageStatus(Message, Status):
26     if Status == 'RECEIVED':
27         uname = Message.FromHandle
28         print '=> received from ' + uname + ': ' + Message.Body
29
30 # If the user sent the passphrase, return the bridge IP's
31         if re.search('.*pass.*', Message.Body):
32             message = 'here are your bridges;\n' + '10.0.0.1:443\n' + '
                  172.16.1.1:443\n' + '192.168.1.1:443\n'
33             skype.CreateChatWith(uname)
34             skype.SendMessage(uname, message)
35             print '<= message sent to: ' + uname + ': ' + message
36             print '=============================================='
37
38 skype = Skype4Py.Skype()
39 skype.OnAttachmentStatus = OnAttach
40 skype.OnMessageStatus = OnMessageStatus
41
42 print('**************************************')
43 print 'Connecting to Skype..'
44 skype.Attach()
45
46 # quits the program if exit is typed into the python console
47 Cmd = ''
48 while not Cmd == 'exit':
49     Cmd = raw_input('')
```

Skype Client

```
1  ##################################################################
2  # marek.kuczynski@os3.nl, december 2010
3  # SSN overlay network project
4  #
5  # This script uses the Skype API to reply bridge addresses to
6  # a user if the correct passphrase is given. It can be extended
7  # in order to retrieve the bridge IP's from a database and to
8  # retrieve the passphrases from another source.
9  ##################################################################
10 import sys
11 import re
12 import os
13 import Skype4Py
14
15 # Check if the Skype client is running and if API communication is possible
16 def OnAttach(status):
17     print 'API attachment status: ' + skype.Convert.AttachmentStatusToText(status)
18     if status == Skype4Py.apiAttachAvailable:
19         skype.Attach()
20
21     if status == Skype4Py.apiAttachSuccess:
22        print('************************************')
23
24 # This routine checks for inbound messages and returns bridge IP's if the correct
      passphrase is given
25 def OnMessageStatus(Message, Status):
26     if Status == 'RECEIVED':
27         uname = Message.FromHandle
28         print '=> received from ' + uname + ': ' + Message.Body
29         skype.CreateChatWith("os3ssn")
30         skype.SendMessage("os3ssn", "pass")
31
32 skype = Skype4Py.Skype()
33 skype.OnAttachmentStatus = OnAttach
34 skype.OnMessageStatus = OnMessageStatus
35
36 print('************************************')
37 print 'Connecting to Skype..'
38 skype.Attach()
```

GPG Encrypt

```
1  #!/bin/bash
2
3  if [[ ${@} < 2 ]]; then
4  echo "Usage: ./`basename $0` <gpg id|Gmail account>" >&2
5  echo "Example: ./`basename $0` 3E1A16CA kdekok@gmail.com" >&2
6  exit 3
7  fi
8
9
10 function GpgRecvKey (){
11 /usr/bin/gpg --recv-key $1 2> ~/tmp/$1.txt >&2
12 if [[ $? != 0 ]]; then
13         echo "Key not found"
14         rm ~/tmp/$1.txt >&2
15         exit 3
16 fi
17 }
18
19 function GpgEncryptFile (){
20 NAME=`egrep  "$2" ~/tmp/"$1".txt | awk -F "\"" '{print $2}' | awk -F "<" '{print $1
       }'`
21 /bin/cp iplist.txt iplist_"$1".txt
22 /usr/bin/gpg --encrypt --recipient "$NAME" iplist_"$1".txt
23 /usr/bin/gpg --delete-key --batch --yes "$2" >&2
24 }
```

GPG Query

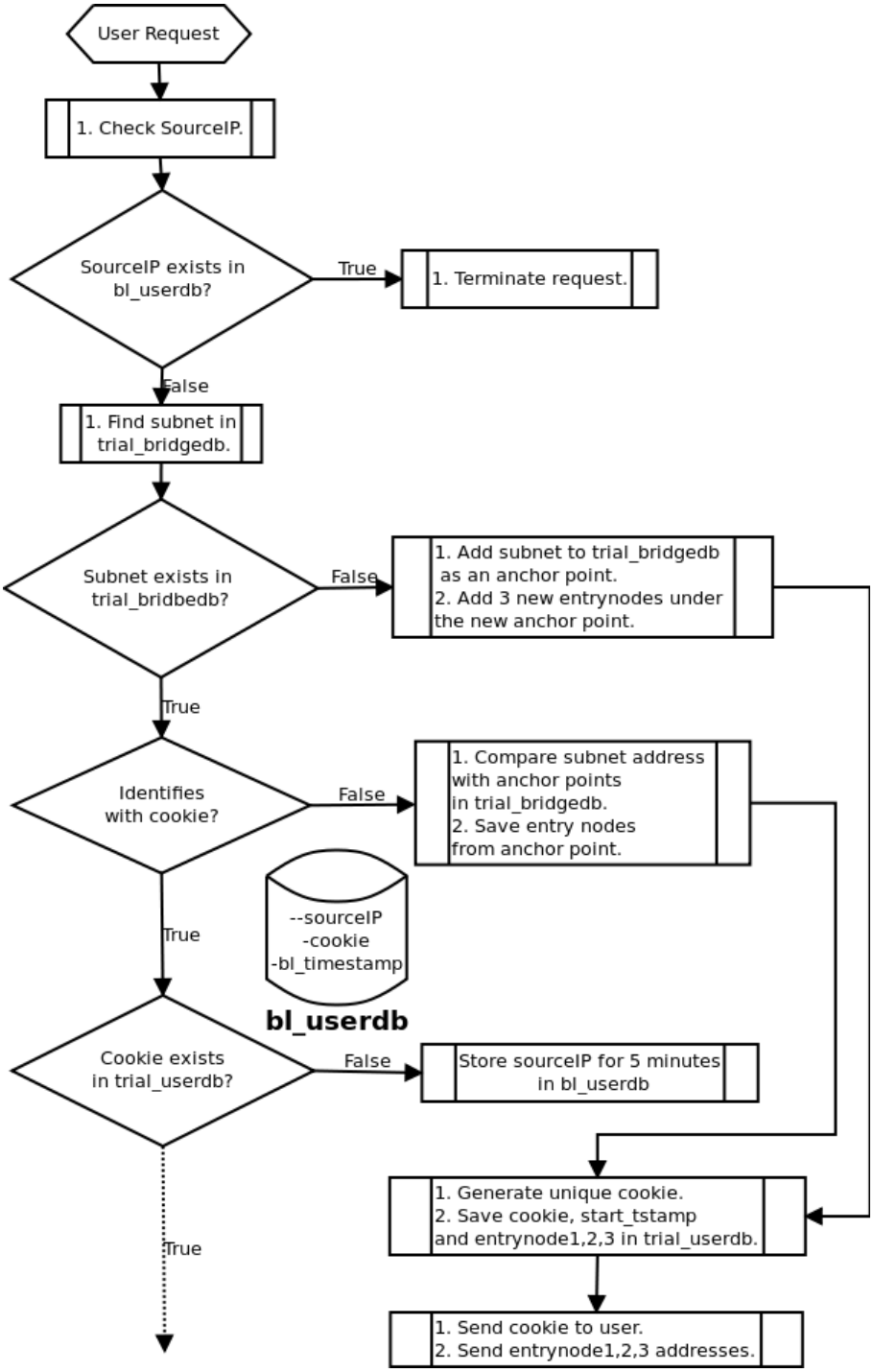
```
#!/bin/bash

# $1 = gpg id
# $2 = email

if [[ ${@} < 2 ]]; then
echo "Usage: ./`basename $0` <gpg id|Gmail account>" >&2
echo "Example: ./`basename $0` 3E1A16CA kdekok@gmail.com" >&2
exit 3
fi


function GpgRecvKey (){
/usr/bin/gpg --recv-key $1 2> ~/tmp/$1.txt >&2
if [[ $? != 0 ]]; then
        echo "Key not found"
        rm ~/tmp/$1.txt >&2
        exit 3
fi
}

function GpgEncryptFile (){
NAME=`egrep  "$2" ~/tmp/"$1".txt | awk -F "\"" '{print $2}' | awk -F "<" '{print $1
    }'`
/bin/cp iplist.txt iplist_"$1".txt
/usr/bin/gpg --encrypt --recipient "$NAME" iplist_"$1".txt
/usr/bin/gpg --delete-key --batch --yes "$2" >&2
}

GpgRecvKey $1 $2
if [[ "$2" == `grep -o "$2" ~/tmp/"$1".txt` ]]; then
        GpgEncryptFile $1 $2
                echo "Hi Anonymous," >> ~/msg_"$1".txt
                echo " " >> ~/msg_"$1".txt
                echo "Hereby the requested Tor entry nodes:" >> ~/msg_"$1".txt
                echo " " >> ~/msg_"$1".txt
                echo "Store the attached file on your local harddrive and decrypt
                    it with the following command:" >> ~/msg_"$1".txt
                echo " " >> ~/msg_"$1".txt
                echo "gpg --output tor_entry_nodes.txt --decrypt iplist_"$1".txt.
                    gpg" >> ~/msg_"$1".txt
                echo " " >> ~/msg_"$1".txt
                echo "Be careful with sharing the nodes!" >>  ~/msg_"$1".txt
                echo "" >> ~/msg_"$1".txt
                echo "~Anonymous~" >> ~/msg_"$1".txt
         mutt -e "set from="do@not.reply"; set realname=Anonymous" -s "Requested
             entry nodes" "$2" -a iplist_"$1".txt.gpg < ~/msg_"$1".txt
        /bin/rm ~/iplist_"$1".txt.gpg
        /bin/rm ~/msg_"$1".txt
        /bin/rm ~/tmp/$1.txt
fi
```

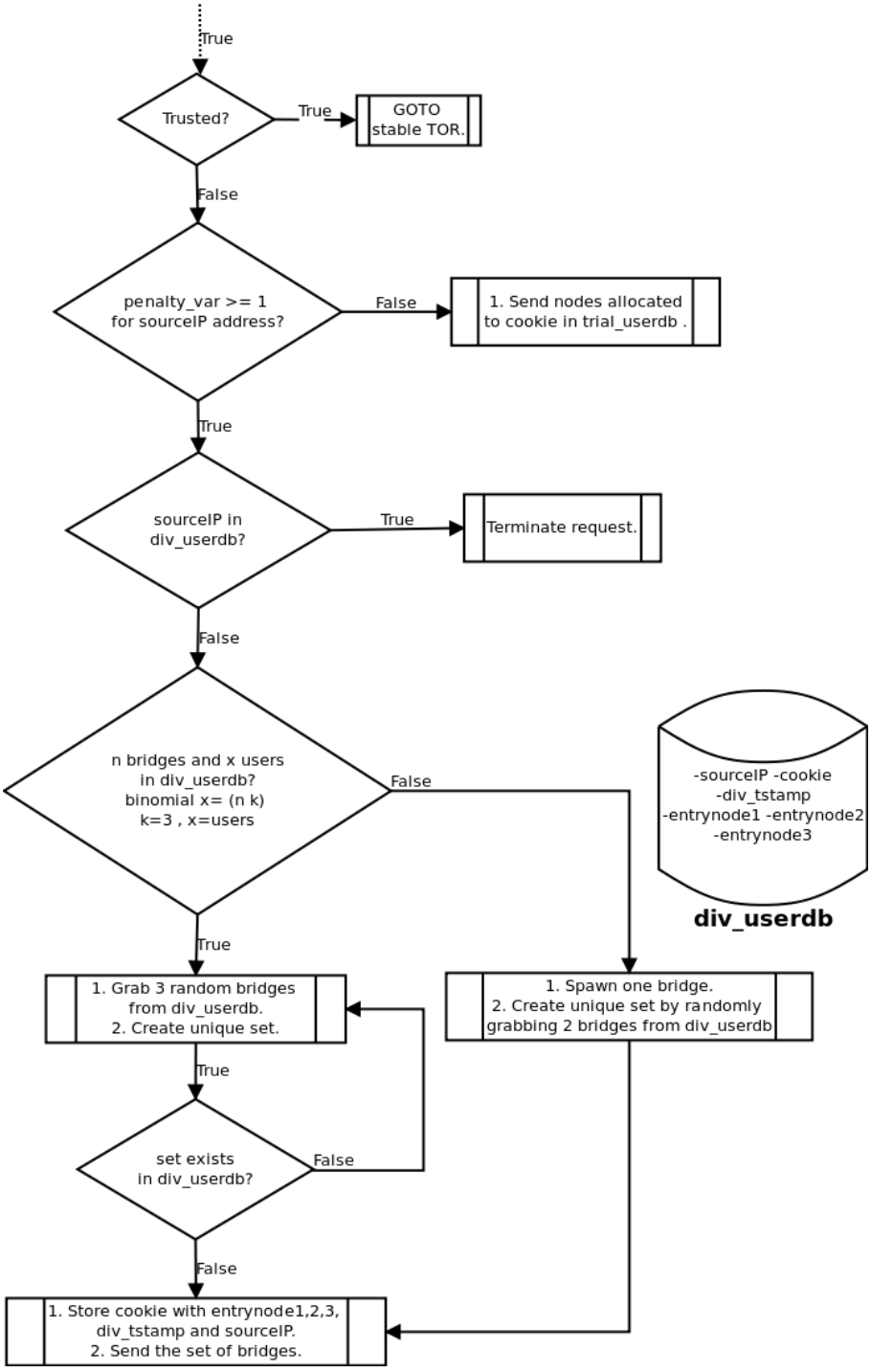
Blocked port

```
1  #!/usr/bin/python
2
3  #No ugly IPv6 warning from scapy
4  import logging
5  logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
6
7  from scapy.all import *
8
9  #Disable annoying output
10 conf.verb=0
11
12 #Create packet, sport <=> Tor dst port
13 #ans, unans = sr(IP(dst='203.208.39.104') /TCP(dport=80, sport=(6523),flags='S',seq
       =1),timeout=1)
14 #Google china
15 ans, unans = sr(IP(dst='203.208.39.104') /TCP(dport=80, sport=(443),flags='S',seq
       =1),timeout=1)
16 #Baidu
17 #ans, unans = sr(IP(dst='220.181.6.175') /TCP(dport=80, sport=(443),flags='S',seq
       =1),timeout=1)
18
19 for x in unans:
20         if x.flags == 0:
21                 print 'Port is blocked'
22
23 for y in ans:
24         if y[0].flags == 0:
25                 print 'Port is unblocked'
```

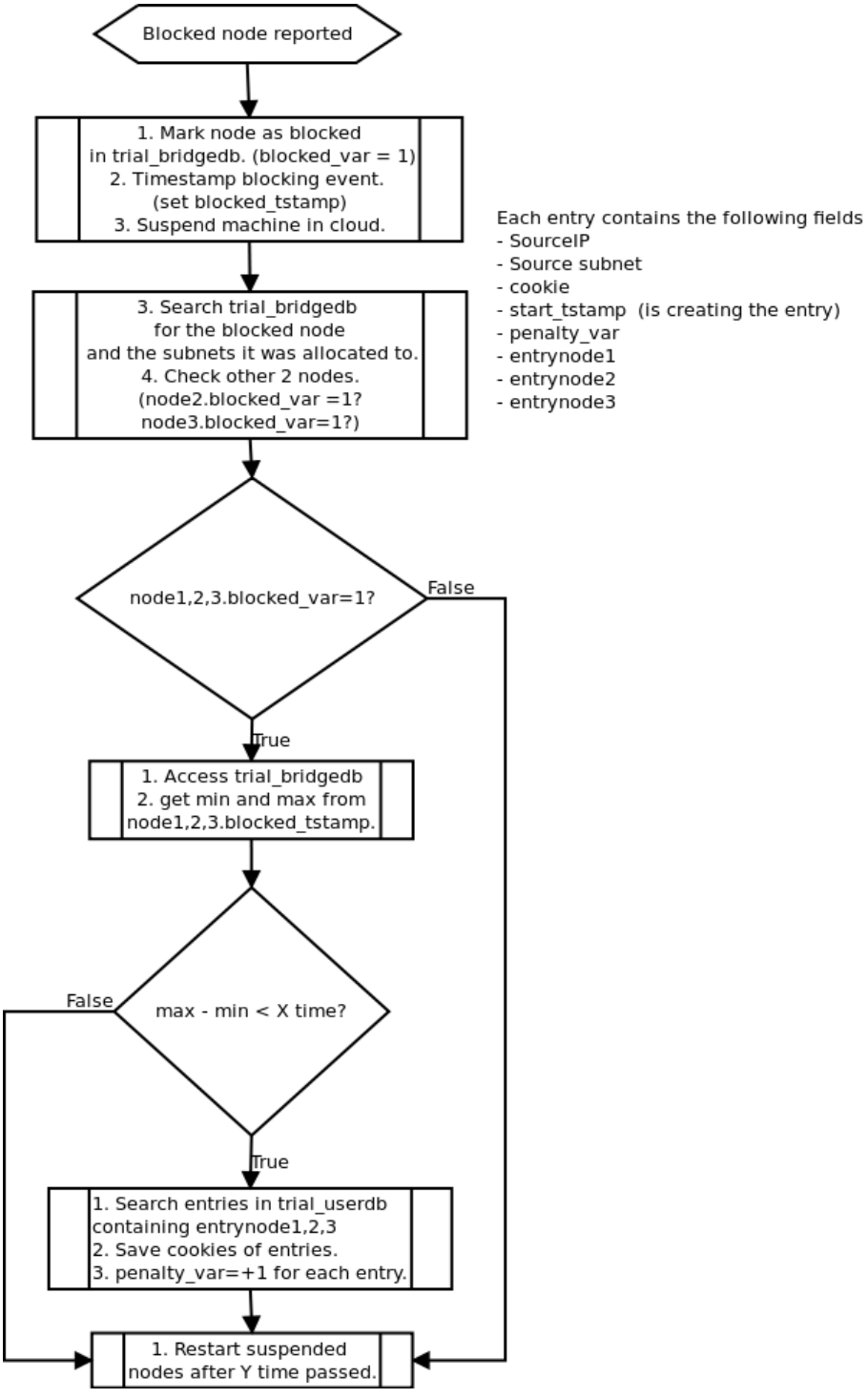
B Reputation trigger event flowcharts

Request event




True
Trusted?
True
GOTO
stable TOR.
False
penalty_var >= 1
for sourceIP address?
False
1. Send nodes allocated
to cookie in trial_userdb .
True
sourceIP in
div_userdb?
True
Terminate request.
False
n bridges and x users
in div_userdb?
binomial x= (n k)
k=3 , x=users
False
-sourceIP -cookie
-div_tstamp
-entrynode1 -entrynode2
-entrynode3
div_userdb
True
1. Grab 3 random bridges
from div_userdb.
2. Create unique set.
1. Spawn one bridge.
2. Create unique set by randomly
grabbing 2 bridges from div_userdb
True
set exists
in div_userdb?
False
False
1. Store cookie with entrynode1,2,3,
div_tstamp and sourceIP.
2. Send the set of bridges.

Blocking event



Reshuffling event

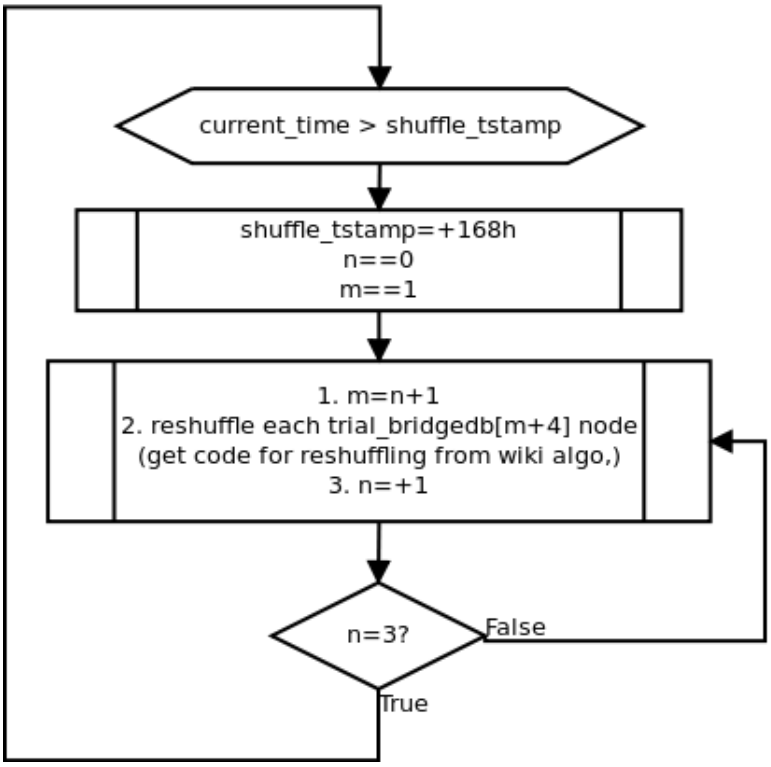
Two types of entries:

Anchor point[subnet_address], stored in 0,4,8,12...
positions of table.
- subnet_address: corresponds to the subnet a client comes from

Entry node[entry_nodeIP, blocked_var, blocked_tmestamp]
stored in 1,2,3, 5,6,7, 9,10,11, positions of table.
- blocked_var: defines if the bridge is blocked
- blocked_tstamp defines time of blocking event.

Other variables:

shuffle_tstamp - establishes time interval
current _time - for checking the time interval
m - for establishing which nodes to reshuffle
n - to limit the number of reshufflings to 3.



References

[1] Tor webpage, *The Onion Routing* http://www.torproject.org/.

[2] Phobos, *Tor partially blocked in China* https://blog.torproject.org/blog/tor-partially-blocked-china.

[3] Roger Dingledine, Nick Mathewson and Paul Syverson, *The Second-Generation Onion Router* https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf.

[4] Jarkko. Oikarinen, *Internet Relay Chat* http://tools.ietf.org/rfc/rfc1459.txt.

[5] Werner Koch, *Implementation of OpenPGP* http://www.gnupg.org/

[6] Callas et al, *OpenPGP Standard* http://tools.ietf.org/rfc/rfc4880.txt

[7] Efnet IRC network http://www.efnet.org/?module=servers

[8] OFTC IRC network http://www.oftc.net/oftc/

[9] Tor blog about China https://blog.torproject.org/category/tags/china

[10] Tor blog about Iran https://blog.torproject.org/category/tags/iran

[11] DomainKeys Identified Mail http://en.wikipedia.org/wiki/DomainKeys_Identified_Mail

[12] Advanced Encryption Standard http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[13] Skype Security http://en.wikipedia.org/wiki/Skype_security

[14] Bouncer (BNC) http://en.wikipedia.org/wiki/BNC_(software)

[15] Google Web Crawler http://en.wikipedia.org/wiki/Googlebot

[16] Tunneling SSH traffic in HTTP(S) traffic http://dag.wieers.com/howto/ssh-http-tunneling/

[17] SP wil Nederlanders niet permanent aftappen http://www.security.nl/artikel/35359/1/SP_wil_Nederlanders_niet_permanent_aftappen.html

[18] Interview with Skype http://bit.ly/grsdEC