Evaluation of the Anonymous I2P Network's Design Choices Against Performance and Security

Juan Pablo Timpanaro, Thibault Cholez, Isabelle Chrisment, Olivier Festor

Evaluation of the Anonymous I2P Network's Design Choices Against Performance and Security

Juan Pablo Timpanaro*, Thibault Cholez*, Isabelle Chrisment*, Olivier Festor*
*TELECOM Nancy, Université de Lorraine
INRIA/LORIA UMR 7503
Vandœuvre-lès-Nancy
F-54602, France
{juanpablo.timpanaro, thibault.cholez, isabelle.chrisment, olivier.festor}@loria.fr

***Abstract*—Anonymous communications are growing extremely fast because more and more Internet users employ anonymous systems, such as the I2P or Tor networks, as a way to hide their online activity. Therefore, these networks have been more and more studied, mainly from a security point of view. Different studies have shown important design flaws in these systems that could break users' anonymity and how these issues can be overcome, but the resilience of the underlying information systems has not been much investigated so far. Indeed, these anonymous systems rely entirely on directories, either centralised or decentralised, to store vital network information.**

In this paper, we consider the I2P anonymous system and its decentralised directory, known as the netDB, where our contributions are twofold. On the one hand, we conduct arguably the first *churn* study of the I2P network, showing that I2P users are more stable than non-anonymous peer-to-peer users. On the other hand, we analyse the design of the netDB and compare it against the popular KAD design, demonstrating that the former is more vulnerable to different attacks, specially to Eclipse attacks, which can be mitigated by applying design choices of the latter. We lately show the positive impact on performances of including KAD's DHT configuration into the netDB in terms of bandwidth, storage and messages overhead.

***Index Terms*—Attack detection, Eclipse Attack, I2P, anonymous systems, netDB**

I. Introduction

ANONYMOUS communications are quickly growing: the Tor network has tripled its user-base in the last six months, while the I2P anonymous network has doubled its user-base in the last ten months[1]. These systems allow users to access different services while preserving their online anonymity, where a user's real identity is decoupled from its assigned system's identity. As with the increased use of these systems, different attacks have been designed and deployed. Regarding the Tor network [1], different studies have been carried out, where the network itself has been attacked [2], [3], or it has been monitored [4]. The I2P network has been attacked as well, at an application level [5] and at a network level [6].

These systems use a directory to store network metadata, *i.e.* the metadata necessary to maintain the operation of the network, such as the information of participants or applications and services deployed within the network. Tor uses a central directory to coordinate its Tor routers, while the I2P network uses a distributed directory to coordinate and store all system metadata.

Our motivation lies in the fact that if these directories are attacked and fail, the entire system will not be able to properly operate and deliver its main service: anonymous communications. In this paper, we focus on the security aspects of the anonymous I2P network and its distributed directory, called the *netDB*.

Our contributions are:

- We evaluate the security and design of the netDB.
- We carry out the first *churn* study of the netDB.
- We improve netDB's design based on lessons learned from the KAD distributed hash table.

The rest of the paper is organised as follows: Section II-D introduces the I2P anonymous network. Section III introduces the related work and arguably the only practical attack against the I2P network. Section IV introduces our recommendations to improve netDB, from a security and performance point of view. Section V introduces the performance evaluation of our proposed mechanisms: extending its replica set size and extending its user-base. Section VI considers further options when dealing with Sybil attacks within distributed hash tables. Finally, Section VII concludes our work and presents further work perspectives.

II. The I2P network

The I2P network is designed as an anonymous network layer enabling users to deploy their own applications on top of the network, and it is mainly designed for anonymous file-sharing and anonymous web hosting. On the contrary to the Tor network, where users' traffic enters the network, gets re-routed and exits to the normal Internet, in the I2P network the traffic stays within the network. Thus, the I2P network is a closed network, on the opposite to the Tor network, that can be seen as an Internet proxy.

A. I2P architecture

Every node, or I2P user, within the system deploys an *I2P router*, thus forming the I2P network overlay. I2P routers connect among themselves forming *tunnels*: a multi-hop path

[1]Statistics from http://metrics.torproject.org and http://stats.i2p.in/, respectively. Last visited on 09/2014.

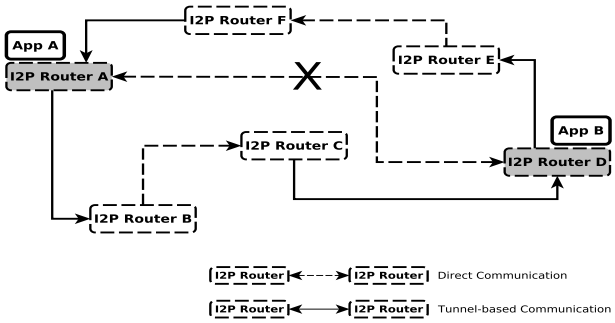

Fig. 1. Architecture of the I2P network

among different I2P routers, as depicted in Figure 1. The `I2P router A` sends messages through a one-hop tunnel employing `I2P router B` (endpoint) and receives messages employing as well a one-hop tunnel using `I2P router F` (gateway). `I2P router D` uses as well one-hop tunnels, where it sends data through the `I2P router E` (endpoint), whilst receives data through the `I2P router C` (gateway). The number of hops in a I2P tunnel varies between 0 and 7, where more hops in a tunnel increases the anonymity but reduces performance, being that data needs to traverse more intermediate nodes.

Users deploy I2P applications on top of their I2P routers, where these applications are able to communicate to other remote I2P applications in an anonymous manner. Figure 1 presents a typical scenario, where applications `App A` and `App B` communicate among themselves by means of these I2P tunnels, avoiding any direct communication among their I2P routers. `App A` uses the tunnel with `I2P router F` as gateway to receive data, while `App B` uses the tunnel with `I2P router C` as gateway to receive data. Similarly, `App A` uses the tunnel with `I2P router B` as endpoint to send data, while `App B` uses the tunnel with `I2P router E` to send data.

Tunnel-based communications is the core of I2P anonymity, where the identity of an I2P user (represented by an I2P router, for example I2P `router A`) is *decoupled* from the identity of an I2P application (for instance, `App A`).

B. I2P anonymity

The information regarding an I2P router is gathered in a structure known as *routerinfo*. That data structure holds all the contact information for that particular I2P router.

An I2P application is not identified through the normal <IP address, port number> tuple, but via a location-independent identifier, known as *I2P destination*. That I2P destination along with a set of encryption keys (to send encrypted data to the application), a signing key and the list of the gateways used to receive data is gathered in a structure known as *leaseset*. Considering the scenario in Figure 1, the leaseset of `App A` contains `I2P router F` as the single gateway, whilst the leaseset for `App B` contains `I2P router C` as the single gateway.

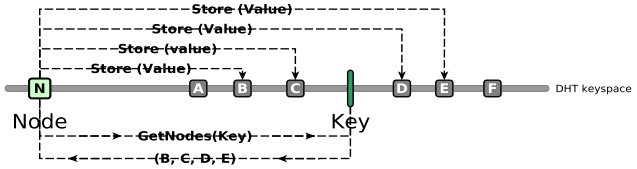

Fig. 2. Kademlia storing/retrieving process

Therefore, an I2P router is identified by a routerinfo, whilst an I2P application is identified by a leaseset. Routerinfos and leasesets are the I2P's network metadata required for a normal network operation. I2P uses a distributed directory to store all I2P metadata and make it accessible to the entire network.

C. Distributed directory

I2P uses a distributed directory to store its network metadata that is composed of leasesets and routerinfos. The database is known as the *netDB* and is a Kademlia-based [7] distributed hash table, composed of *floodfill* nodes. Floodfill nodes are normal I2P routers with high bandwidth rates. This means that the netDB is not formed by the entire network, as in KAD, but only by a subset of all I2P routers.

The netDB works as any Kademlia-based distributed hash table when storing and retrieving data. Figure 2 presents a simplification of this procedure, where a node `N` stores data with a particular `key` value. The first step is to retrieve those nodes close to the key value, known as *replica set*, in this case nodes `B`, `C`, `D` and `E` (considering a replica set of four nodes, as an example). The second and last step is to send the *store* message (in the case of searching for data, this will be a *search* message) to those nodes, thus storing the value.

The netDB works in a similar manner, where every floodfill node stores a portion of all network metadata, either leasesets or routerinfos.

D. Keyspace shifting

As every Kademlia-based DHT, I2P's netDB uses the XOR metric to determine in which nodes a value should be published in or retrieved from by comparing a node's identifier and a value's identifier. A node's identifier is determined during the first execution and normally remains unchanged throughout the entire life of the node.

Nevertheless, the netDB uses a *temporary identifier* instead of a fix identifier to compute the XOR distance. This temporary identifier, called a *routing identifier* as opposed to the *node identifier*, is obtained by appending the node identifier with the current date and hashing the result, as shown in Equation 1. Therefore, the identifier used in the netDB is the routing identifier, which changes every day, while the node identifier remains fix.

$$routing\_id = SHA256(node\_id || yyyyMMdd) \quad (1)$$

At midnight, every previously published value needs to be republished in another DHT location, since the routing

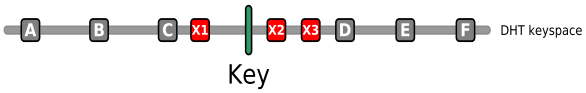

Fig. 3. Egger *et al.* Eclipse attack on the netDB

identifier changes. I2P uses this approach to increase the cost of a localised Sybil attack [8]. During a Sybil attack, an attacker creates several fake identities in order to place them around the DHT. In a localised Sybil attack, all these fake identities are place close to a particular target, as to gain control of a particular portion of the DHT.

III. Related work

The Tor and I2P networks are arguably the most widely deployed low-latency anonymous systems.

Different attacks have been deployed within the Tor network, aiming at disrupting the network or degrading the anonymity of the system [9], [3]. Monitoring the system has as well been studied [10], [4], were different insights about Tor users have been obtained.

Regarding the I2P network, Herrmann *et al.* [6] conducted an attack against I2P's peer selection mechanism [11], de-anonymising I2P anonymous web sites, known as *eepsites*. Crenshaw [5], on the other hand, exploited I2P's application layer to link together eepsites and the I2P users running those eepsites, thus de-anonymising I2P users. On the contrary to the Tor network, monitoring the I2P network has not been widely studied.

Despite these previous studies, none of them considered the network directory. The central directory of the Tor network has been, however, questioned about its centralised nature in [12], where a decentralised hash table has been considered as an alternative solution. Regarding the I2P network, Egger *et al.* [13] conducted arguably the solely practical attack against the I2P network. The authors deployed an attack against the netDB, aiming at eclipsing a particular leaseset within the netDB. As mention in Section II-D, a leaseset identifies a particular application, such as a file-sharing client. The goal of Egger *et al.* attack is to render completely inaccessible to the rest of the network the leaseset identifying an application. The most similar scenario in the normal Internet will be to erase the DNS entry of a particular website.

Figure 3 illustrates Egger *et al.*'s attack. The goal is to place malicious peers around a particular target key, such as `X1`, `X2` and `X3`, closer than any other legitimate peer, such as `C` or `D`. Thus, these malicious peers will store that target key and eventually they will stop answering request, eclipsing the key from the rest of the network.

In order to deploy this attack, Egger *et al.* need to brute-force the position of these malicious peers, due to the key shifting mechanism used by the netDB. The number of generated malicious peers or *fake keys* is proportional to the size of the replica set used within the netDB, as later detailed in Section IV-C: a larger replica set requires a larger set of fake keys to be generated and inserted. We will consider the attack by Egger *et al.* in our study and analyse how the cost of this attack can be increased, where we additionally consider a mechanism to detect whether a leaseset is under attack.

IV. Improving the resilience of the *netDB*

I2P's reduced replica set has a negative impact from two points of view: From a reliability point of view, a low replica set along with a considerable churn value in the network can produce data loss [14]. From a security point of view, an attacker will need to generate fewer fake routing keys so as to conduct an attack, specially localised Sybil attacks, which are the basic attack to deploy further complex attacks, such as Eclipse attacks. A lower replica set means that an attacker will have less *competition* with regular peers, which is translated to a lower attack cost. So as to increase competition we can:

- Increase the replica set size.
- Increase the number of nodes supporting netDB.

This section first introduces the KAD network. Later, it introduces two significant improvements for the netDB: extension of its replica set and extension of its user-base, following KAD's design.

A. The netDB and the KAD network

The KAD network is a Kademlia-based distributed hash table used by two file-sharing clients, aMule and eMule. It has a double-indexation level, where the first level indexes content, whilst the second indexes available sources, serving as a fully decentralised search engine and tracker. The netDB, on the contrary, has a single-indexation level, where peers and application metadata are indexed, similar to KAD's second level of indexation.

In both networks, routing is iterative, *i.e.* a peer iteratively searches for peers close to a particular key, before sending a service message, such as a `store` or `search` message. The difference, however, is the size of the *replica set*. When a peer needs to store a value within KAD or the netDB, he needs first to locate the closest peers to the key, before issuing the service message. This set of closest peers is called the replica set for that particular key. In a network with no churn, the replica set has ideally a single peer on it. However, being that churn in an intrinsic property of P2P networks, both KAD and the netDB have multiple peers in the replica set. KAD, on the one hand, uses a replica set of ten peers and therefore every time a peer needs to store a value, it is replicated in the ten closest peers. The netDB, on the other hand, only uses the three closest peers to replicate a particular value.

It is important to mention that the replica set size of the netDB has been decreased, from eight to five to finally three nodes, due to our previous monitoring study [15], where we monitored the network based on a distributed architecture to characterise users and anonymous applications. We considered the distribution of our monitor nodes, reaching the conclusion that we needed $\lceil N / X \rceil$ monitor nodes to gather all network metadata simultaneously, being `N` the size of the netDB (total number of floodfill nodes) and `X` the size of the replica set. For example, considering `N`=1000 and `X`=8 nodes we needed

⌈ 1000 / 8 ⌉= 125 monitor nodes to have a complete view of the netDB. However, considering a lower replica set of three nodes, we needed ⌈ 1000 / 3 ⌉= 334 monitor nodes. I2P's designers considered our study and decided to reduce the replica set[2] to tamper our monitoring efforts.

However, this was clearly a poor design choice for two reasons. On the one hand, as opposed to the KAD network where requests are directly addressed and unencrypted, there is no privacy issues when monitoring the I2P network. Requests to the netDB are tunnelled employing the same tunnels as for anonymous communications, as explained in Section II-A, and therefore an attacker can not link together an application (defined by a leaseset) and the final user requesting this application. On the other hand, reducing the replica set makes the I2P network more vulnerable to Sybil attacks, and therefore to massive DoS, while not improving its privacy nor its security.

B. Increasing the replica set

We aim at increasing the effort to deploy a Sybil attack, that is to be able to position malicious peers next to a particular key, closer than any other legitimate peer. The current netDB design considers a replica set of three peers, therefore an attacker needs to compute three fake routing keys closer than any other peer to achieve a successful attack. The netDB's design does not allow to choose a peer's position within the netDB space and therefore an attacker needs to compute fake routing identifiers in a brute-force manner to place its malicious peers.

Considering a netDB with `N` floodfill nodes and a random routerinfo (or leaseset) with key `Kr`, the average number of shared bits between the key `Kr` and the **closest** floodfill `F` is $nb\_bits = \log_2 N$. That is, if we consider the current size of the netDB ($\sim$ 4000 nodes) and a random routerinfo, or leaseset, we will find that the closest floodfill share 12 bits in common with the key `Kr` associated to that routerinfo or leaseset.

Now, considering the previous relation between the number of users in the netDB and the number of shared bits, Equation 2 depicts the number of *fake routing keys* an attacker will need to generate based on the size `K` of the replica set. Considering netDB's current configuration with `K` = 3 and `N` $\sim=$ 4000 , an attacker will need to generate in average more than 12K fake routing keys before finding the number of appropriate ones to perform the attack.

$$\texttt{fake\_routing\_keys} = \texttt{K} \times 2^{\lfloor \log_2 N \rfloor + 1} \quad (2)$$

We additionally conducted an experiment considering the real netDB, so as to complement our analytic results. In order to correctly measure the effort to compute these fake routing keys, we conducted the following experiment, simulating an attack:

- Step 1: we published close to 200 keys in the netDB and stored the replica set for each publication.

[2]http://zzz.i2p/topics/1281. Last visited on 09/2014.

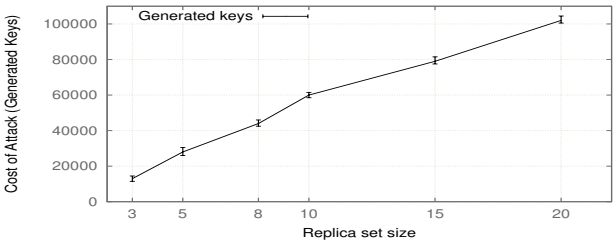

Fig. 4. Cost of Attack/Replica Set size ratio

- Step 2: for each publication we computed the closest peer, within the replica set, to the published key.
- Step 3: finally we computed, in a brute-force manner, routing keys until obtaining three routing keys closer than the closest legitimate peer computed in step two. We repeated this step several times (over a hundred times).

Thus, we can measure how many fake routing identifiers an attacker need to generate before being able to fully eclipse a particular key in the netDB. In other words, which is the **cost of the attack**, measured in *generated fake routing keys*. This behaviour can be observed in Figure 4, where for a replica set value of three, we need to generate, in average, 13000 fake routing identifiers, where with moderate resources, can be computed in a few minutes. We obtained these values within the netDB containing approximately 4000 nodes.

We performed the same experiment with different configurations of replica set, as observed in Figure 4, for values 3, 5, 8, 10, 15 and 20 nodes. By increasing the replica set of the netDB, we linearly increase the effort of the attack. Considering the replica set of KAD for instance, an attacker will need to generate up to 60000 fake routing identifiers before obtaining a valid set of malicious peers.

Increasing the replica set within the netDB has two effects. On the one hand, we are hardening localised Sybil attacks on the network, a major problem in I2P's threat model[3]. On the other hand, increasing the replica set increases network resilience against churn, since the probability of all peers in the replica set going offline in a particular period of time exponentially decreases when the size of the replica set increases, as detailed in Section V-C.

C. Increasing netDB's user-base

The netDB design states that only certain qualified peers can join the network, mainly filtering out peers regarding their bandwidth limits and their uptime in the system. I2P's designers states that, in this way, the netDB is formed by fast and stable peers, the *floodfill nodes*. Although a valid approach, we argue that these limitations need to be removed from the netDB design, evolving to a more open design, such as KAD. In KAD, every reachable peer supports the DHT, without any restriction, leading to a network size of several millions peers [16]. KAD deals with less stable peers and churn

[3]https://geti2p.net/en/docs/how/threat-model. Last visited on 09/2014.

by increasing its replica set and adjusting its republication window accordingly.

Increasing the number of peers within the netDB implies that an attacker will need to generate further fake routing keys, since the netDB space will be more dense. Table I presents our results of the cost of an Eclipse attack based on a given replica set and netDB size. With the current values of replica set (three nodes) and the netDB size (~ 4000 nodes), an attacker needs to generate approximately 13000 fake routing keys to achieve a full Eclipse attack, which matched our previous analytic results. However, if we allow every node in the I2P network (~ 55000 nodes) to join de netDB and maintain the current replica set of three nodes, the cost of the attack grows almost by a factor of ten, where an attacker will need to generate over 123000 fake routing keys.

We computed the current number of floodfill nodes and the estimated number of users in the I2P network based on our distributed architecture [15].

Replica Set	NetDB Size			
	1X ($\sim 4K$)	2X	4X	All ($\sim 55K$)
Size= 3	**13K**	27K	52K	123K
Size= 5	23K	65K	110K	224K
Size= 10	60K	125K	250K	**730K**

TABLE I
NUMBER OF ATTACK KEYS TO GENERATE, BASED ON THE REPLICA SET AND NETDB SIZE

We consider the DHT configuration of KAD for two reasons. On the one hand, the KAD network has been widely study and improved [17], [18], [19], [20], [16] over the years, yielding to a mature and robust design. On the other hand, it has been in use for more than ten years, serving as the backbone of the widely known eMule and aMule file-sharing clients, proving to be an excellent distributed directory to support a large-scale file-sharing network.

Increasing the replica set to ten peers to match KAD's DHT configuration and letting every I2P node to become part of the netDB will increase the cost of an Eclipse attack by a factor of fifty (from 13K to 730K), where an attacker will now need to generate almost a quarter of a million fake routing keys to achieve the same attack. The cost of the attack now will remain proportional to the size of the network, which is still constantly growing. Extrapolating our results from Table I, we can estimate that an attacker will need to generate over 15 million fake routing keys considering a network with one million users. If we consider that the non-anonymous KAD network has approximately three million concurrent online users [16] and the anonymous Tor network has an user base of over two million users [4], we can think in a one-million users I2P network as a possible scenario.

Increasing the size of the netDB consists in allowing every peer to automatically form part of the system, as in KAD. The current analysis of bandwidth and uptime should be removed from the current client, since the load of maintaining the netDB in terms of bandwidth is negligible when compared with the bandwidth assigned to I2P routers, as we will see in Section V-B. Moreover, the uptime of nodes is no longer an issue, since a replica set of ten peers reduces the probability of loosing all nodes in the replica set to almost zero, as shown in Section V-A.

[4]Statistics from https://metrics.torproject.org/users.html. Last visited on 09/14.

V. PERFORMANCE EVALUATION

This section first introduces a *churn* analysis of the I2P network, which will assist us in our performance evaluation later in this section. We conduct a performance evaluation from two points of view: the cost of increasing netDB's user base and the cost of increasing netDB's replica set.

A. Churn in the I2P network

The I2P network is a peer-to-peer network where I2P users interconnect among themselves to create multi-hop paths or *tunnels*, creating an overlay on top of the normal Internet. As with every widely-deployed public peer-to-peer network, it suffers from variation on the number of stable users in the system, also known as *churn* or *dynamics of peer participation* as defined by Stutzbach and Rejaie [14], among others.

Stutzbach and Rejaie conducted a comprehensive study on network churn in P2P systems, yielding to conclude that most P2P systems, including the KAD network, can be characterised through a log-normal distribution function, as shown in Figure 5. According to Stutzbach and Rejaie, the probability of a node going offline, let's call it $\mathrm{p}_{\mathit{off}}$, after half an hour oscillates around $\mathrm{p}_{\mathit{off}} = 1 - 0.59 = 0.41$, meaning that with a replica set of three peers and the current I2P republication windows of thirty minutes, we might have, following Stutzbach and Rejaie results, a probability of $\mathrm{p}^3_{\mathit{off}} = 0.41^3 = 0.068$ that the entire replica set goes offline and therefore the value previously stored within those peers becomes inaccessible.

However, we consider that *churn* within the I2P network might not be fully characterised by non-anonymous peer-to-peer networks. Even if the I2P network is mainly used for anonymous file-sharing and anonymous web-hosting, we cannot generalise the dynamics of peer participation of anonymous networks based on the dynamics of simple file-sharing networks, such as KAD or BitTorrent. Therefore, we conduct arguably the first churn study in anonymous networks, in our case the I2P network.

We placed a set of eight floodfill nodes, which gathered routerinfos and leasesets as part of the normal netDB procedure. We ran our floodfill nodes for five days, where we collected, in average, 1424 distinct routerinfos every 24 hours. As mentioned in Section II-C, the netDB's keyspace switches at midnight, meaning that a floodfill will not necessary always store the same routerinfos. Our floodfill nodes stored a total of 49563 different routerinfos during the five-days test period. We computed the session length of every seen I2P user, represented by a single routerinfo. Figure 5 depicts our results as well as Stutzbach and Rejaie past results.

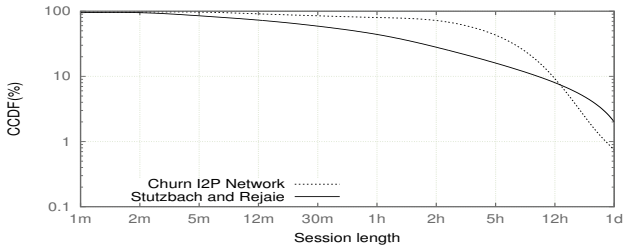

Fig. 5. Measured *churn* in the I2P network

We can observe that $\mathtt{p}_{off}$ is considerable higher, meaning that I2P users are more stable than in a normal non-anonymous peer-to-peer network up to approximately twelve hours, where both curves cross each other. Afterwards, we detect a considerable low number of I2P users with session length superior to twelve hours, as opposed to normal peer-to-peer networks, where more users might rest connected for long periods of a day or more.

In this case we have that the probability $\mathtt{p}_{off}$ of a node going offline after half an hour oscillates around $\mathtt{p}_{off} = 1 - 0.85 = 0.15$, approximately three times less than in a normal peer-to-peer network. Moreover, $\mathtt{p}_{off}$ of the entire replica set is now $\mathtt{p}^3_{off} = 0.15^3 = 0.0033$. Considering our motivation to extend I2P's replica set size to ten peers, we will have $\mathtt{p}^{10}_{off} = 0.15^{10} =\sim 0$.

Our results bring forward an important difference between anonymous users and non-anonymous ones. Anonymous users, in this case I2P users, are more stable in the short run, where the probability of the length of the session exponentially decreases when we pass the two-hour barrier. On the contrary, non-anonymous users present, as depicted by Stutzbach and Rejaie, a log-normal distribution function, where it is not uncommon to observe users connected for long periods, well beyond over 24-hours.

B. Cost of increasing netDB's user-base

The current design of the I2P network states that in order for a node to become a floodfill peer, *i.e.* become part of the netDB, it needs to have a certain bandwidth available and a certain uptime. Currently, a node needs to have more than 128 KBps of available bandwidth and a minimum uptime of 200 minutes in order to become a floodfill node. It becomes logical that a node needs more than three hours of uptime since the size of the replica set is formed by only three nodes and considering unstable nodes will lead to data loss.

We argue that this analysis has become irrelevant for two reasons: nowadays available bandwidth and the minimal memory consumption a node needs when having a floodfill status are negligible.

Bandwidth analysis: On the one hand, according to I2P's designers[5] the bandwidth a floodfill node needs (F_BW) to maintain its floodfill status and to answer remote queries can be computed as shown in Equation 3, where `N` is the average number of users in the I2P network, `L` is the average number of client destinations (leasesets), `F` is the tunnel failure percentage, `R` is the tunnel rebuild period, `S` is the average netDB entry size (either a leaseset or a routerinfo) and `T` is the tunnel lifetime. Although these values are proposed by the designers, they are suitable for an roughly estimate on the required bandwidth.

$$\mathtt{F\_BW} = \mathtt{N} * (1+L) * (1+F) * (1+R) * \mathtt{S}/\mathtt{T} \quad (3)$$

According to I2P's designers, `L` has value of 3, `F` has a value of 0.05, `R` has a value of 0.02. `S` has an average value of 1 KB and finally `T`, the lifetime of a tunnel, is 10 minutes. Considering the current size of the I2P network, which is approximately 55000 users, we can determinate that a floodfill node will need approximately an extra of 0.38 MBps, according to our estimations. This floodfill bandwidth will be taken out from the overall I2P user bandwidth.

According to our previous monitoring study on the I2P network [15], The United States, Russia, France and Germany are the top four I2P detected countries, adding up to almost the 60% of all I2P users. Considering the worldwide bandwidth study carried out by Ookla[6], these four countries have an average bandwidth, or **line speed**, of 3.45 MBps.

I2P guidelines[7] suggest to set the overall I2P user bandwidth slightly under the current line speed of the user. Considering an 80% of the line speed seems reasonable, resulting in approximately 3.45 MBps * 80% = 2.76 MBps for the I2P user overall bandwidth. Thus, a floodfill node will barely require 0.38 MBps * 100 / 2.76 MBps = 14% of the overall available bandwidth, which is completely affordable.

Memory consumption analysis: The extra memory consumption a floodfill node needs to maintain netDB's data is also very small. We monitored our local floodfill node to determine the required storage based on the number of routerinfos and leasesets kept. After 48 hours, where the floodfill is well integrated into the netDB, it keeps a total average of 35 leasesets and 1966 routerinfos. We computed an average leaseset size of 1068 bytes and an average routerinfo size of 849 bytes, where give us a total consumption of 35 * 1068 bytes + 1966 * 849 bytes = 1.62 MB. We consider that this cost is completely affordable as well.

Therefore, a floodfill node will need approximately an extra 0.38 MBps of bandwidth and 1.62 MB of memory, which is, for nowadays computer configurations, almost negligible.

C. Cost of increasing replica set size

Increasing the replica set involves replicating a value to be stored in more nodes and therefore sending more messages. The number of required messages can be considered as $\mathtt{msgs} = rm+sm$, where the former correspond to routing messages, while the later to service messages. Routing messages do not depend on the size of the replica set on Kademlia-based

[5]https://geti2p.net/en/docs/discussions/netdb. Last visited on 09/14.

[6]http://www.netindex.com/. Last visited on 09/14.

[7]http://echelon.i2p.us/I2Pguide/bandwidth.html. Last visited on 09/14.

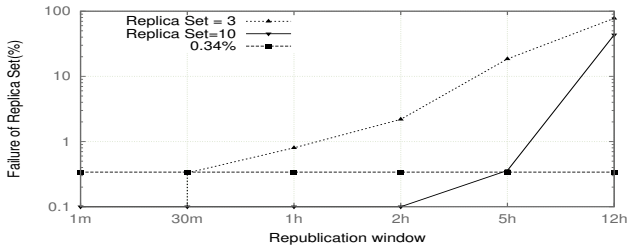

Fig. 6.

networks, while service messages are directly proportional to the size of the replica set. If we consider the republication time window in the netDB, we can determine the number of messages to be sent to store a value in the netDB.

Currently there is a thirty-minutes time window for the republication of values, which means that every half an hour a node needs to send out *rm*+*sm* messages to get a value store in the netDB. If we consider a replica set of ten peers, that is 10/3 times bigger than the current replica set, where the same node will need to send $10/3*(rm+sm)$ messages. Considering that routing messages are not affected by the size of the replica set, we actually have a total of $\mathtt{msgs} = rm+(10/3*sm)$ messages sent every thirty minutes to have a value store in the netDB when employing a replica set of ten nodes.

However, since a larger replica set decreases the overall probability of the entire replica set going offline, a node can afford to extend its republication time window. Figure 6 depicts the probability of the entire replica set going offline after a defined amount of time, for a replica set of three and ten peers. These results were obtained following our previous results, as presented in Section V-A.

As observed in the figure, for a replica set of three nodes, the probability of the entire replica set going offline after half an hour is $\mathtt{p}^3_{off} = (1-0.85)^3 = 0.15^3 = 0.0033$. Considering our new replica set of ten peers and aiming at maintaining the same $\mathtt{p}_{off}$ probability, *i.e.* the same service quality, we have the same probability in a five hours period, as in $\mathtt{p}^{10}_{off} = (1-0.43)^{10} = 0.57^{10} = 0.0036$. Therefore, we can extend the republication of values within the netDB to every five hours, reducing by a factor of three the amount of messages a node needs to send to maintain a value stored.

By extending the republication window to five hours, we maintain the same *service quality* of the current netDB. Additionally, we reduce by a factor of three the number of messages a node needs to send to maintain a value stored within the netDB, since the node will now need to send 10 messages in a five-hours period (2 messages per hour) instead of 3 messages in a thirty-minute period (6 messages per hour previously).

Finally, we do not consider here the overhead of the routing table management induced by the increased user's base. In fact, structured P2P network have for long been proven very efficient at this regard and perform routing in log(N) hops.

VI. Discussion

Extending the size of netDB's replica set as well as increasing the overall user-base of the netDB certainly tamper and attacker's effort to deploy a Sybil attack. However, these mechanisms are only a partial solution to this widely know problem. Urdaneta *et al.* [21] presented a comprehensive study on mechanisms to deal with Sybil attacks, classifying them into different groups: centralised certification entities, distributed mechanisms, identification schemes based on physical network characteristics, social-based approaches, computational puzzles and game-theory approaches. Game theory-based approaches [23], [24] pretend to impose a utility model, which often needs to use a sort of a currency in the system. Since the utilisation of a money currency within a distributed environment is highly complex, these approaches have remained theoretical and have not been deployed in any large distributed system.

Computational puzzles, also introduced as *Resource testing* by Levine *et al.* [22], aim at proving network nodes ID with highly-demanding computational puzzles, assuming that an attacker has not enough computational resources to maintain an artificially high number of Sybil nodes, since several logical identities are dependent on the same physical entity. However, the computational puzzles work best if all identities are proven in a simultaneous way as stated by Douceur [8], and it requires that honest nodes continue to compute these puzzles in a regular basis in order to remain on the system and avoid the collection of identities with time. Given the very high amount of IDs that must be generated to perform a successful eclipse attack when considering the netDB parameter we recommend, relying on crypto-puzzle clearly offers the best collaboration with our work in order to further improve I2P security against eclipse attacks.

A bullet-proof solution against Sybil attacks can only be achieved by means of a centralised approach, as early stated by Douceur [8], while non-centralised approaches can partially deal with this attack. Of course, a distributed approach seems to better fit a DHT than a central authority but, due to the lack of mutual trust in a distributed environment and to the reduced view of the network, as stated by Urdaneta *et al.*, these defence mechanisms are unable to fully prevent a Sybil attack, but only mitigate its effects. Our analysis and recommendations regarding the I2P's netDB brings the network a step closer to a robust system, while maintaining backward compatibility.

VII. Conclusions and future work

In this paper, we evaluated I2P's distributed directory, known as the netDB, from a security and design points of view. From a security point of view, we particularly considered the Sybil attack, which enables further complex attacks, such as a full Eclipse attack, and propose different mechanisms to harden the netDB based on the design of the KAD network. An attacker will now need to compute 50 times more fake identifiers than before to deploy the same attack, considering a replica set of ten peers and the current I2P network size. However, a resourceful attacker can still deploy a full Eclipse

attack on the netDB. Therefore, our solution would gain to be associated with a more resource-consuming way to generate routing ID, by involving crypto-puzzles in the process rather than simple hash functions.

We additionally conducted the first *churn* study the I2P anonymous network, where we detect a slightly different behaviour of anonymous users when compared to normal file-sharing networks. I2P anonymous users keep connected longer hours, up to nine hours, where we found that the session length curve dramatically drops.

Our solution is favourable in every sense: we have a more robust network, which as well reduces message overhead thus improving network overall performances. Even more, our solution can be locally implemented allowing a fully backward compatible I2P client and letting the I2P network to progressively evolve to a safer state.

Our future work consists in a *flexible* and *iterative* approach to overcome this resourceful attacker. The owner of a leaseset, for instance, will periodically search for his published leasesets and if not found, he will republished it using a bigger replica set, thus avoiding malicious peers and reaching out for legitimate peers. This approach enables to *auto-detect* an ongoing attack and adjust the publication procedure accordingly. Another direction that could directly benefit from our present contribution and better mitigate Sybil attacks would be to change the way DHT's identifiers are computed in order to need more resources.

REFERENCES

[1] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium*, SSYM '04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[2] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for tor hidden services: Detection, measurement, deanonymization. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 80–94, Washington, DC, USA, 2013. IEEE Computer Society.

[3] Tor Weekly News. Tor and the RELAY EARLY traffic confirmation attack. https://blog.torproject.org/blog/tor-weekly-news-%E2%80%94-august-6th-2014.

[4] Damon Mccoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining Light in Dark Places: Understanding the Tor Network. In *Proceedings of the 8th Privacy Enhancing Technologies Symposium*, PETS '08, Leuven, Belgium, 2008.

[5] Adrian Crenshaw. Darknets and hidden servers: Identifying the true IP/network identity of I2P service hosts. In *Black Hat DC 2011*, Black Hat '11, DC, March 2011.

[6] Michael Herrmann and Christian Grothoff. Privacy-implications of performance-based peer selection by onion-routers: a real-world case study using I2P. In *Proceedings of the 11th International Conference on Privacy Enhancing Technologies*, PETS '11, pages 155–174, Berlin, Heidelberg, 2011. Springer-Verlag.

[7] Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Revised Papers from the 1st International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, 2002. Springer-Verlag.

[8] John R. Douceur. The Sybil Attack. In *Revised Papers from the 1st International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, 2002. Springer-Verlag.

[9] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-TIN. How much anonymity does network latency leak? *ACM Trans. Inf. Syst. Secur.*, 13(2):13:1–13:28, March 2010.

[10] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. A case study on measuring statistical data in the tor anonymity network. In *Proceedings of the 14th International Conference on Financial Cryptograpy and Data Security*, FC'10, pages 203–215, Berlin, Heidelberg, 2010. Springer-Verlag.

[11] zzz and L. Schimmer. Peer proling and selection in the I2P anonymous network. In *Proceedings of PET-CON 2009.1*, March 2009.

[12] Tor Stack Exchange. DDoS attack on Tor directory. bit.ly/1iWOrmC. Last visited on 12/2013.

[13] Christoph Egger, Johannes Schlumberger, Christopher Kruegel, and Giovanni Vigna. Practical attacks against the i2p network. In *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2013)*, October 2013.

[14] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 189–202, New York, NY, USA, 2006. ACM.

[15] Juan Pablo Timpanaro, Isabelle Chrisment, and Olivier Festor. A Bird's Eye View on the I2P Anonymous File-sharing Environment. In *Proceedings of the 6th International Conference on Network and System Security*, NSS '12, Wu Yi Shan, China, 2012.

[16] Thibault Cholez, Isabelle Chrisment, Olivier Festor, and Guillaume Doyen. Detection and mitigation of localized attacks in a widely deployed P2P network. *Peer-to-Peer Networking and Applications*, 6(2):155–174, May 2012.

[17] Moritz Steiner, Taoufik En-najjary, and Ernst W. Biersack. Exploiting kad: Possible uses and misuses. *ACM SIGCOMM CCR*, 37, 2007.

[18] Peng Wang, James Tyra, Eric Chan-Tin, Tyson Malchow, Denis Foo Kune, Nicholas Hopper, and Yongdae Kim. Attacking the kad network. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, SecureComm '08, pages 23:1–23:10, New York, NY, USA, 2008. ACM.

[19] Moritz Steiner, Ernst W Biersack, and Taoufik En Najjary. Actively monitoring peers in KAD. In *Proceedings of the 6th International Workshop on Peer-to-Peer Systems, February 26-27, 2007, Bellevue, USA*, IPTPS 2007, Bellevue, UNITED STATES, 02 2007.

[20] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Evaluation of Sybil Attacks Protection Schemes in KAD. In Ramin Sadre and Aiko Pras, editors, *3rd International Conference on Autonomous Infrastructure, Management and Security - AIMS 2009*, volume 5637 of *Lecture Notes in Computer Science*, pages 70–82, Enschede, Netherlands, 2009. University of Twente, Springer.

[21] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. A survey of DHT security techniques. *ACM Comput. Surv.*, 43(2):8:1–8:49, 2011.

[22] Brian Neil Levine, Clay Shields, and N. Boris Margolin. A Survey of Solutions to the Sybil Attack. Tech report 2006-052, University of Massachusetts Amherst, Amherst, MA, October 2006.

[23] Rupert Gatti, Stephen Lewis, Andy Ozment, Thierry Rayna, and Andrei Serjantov. Sufficiently secure peer-to-peer networks. *Workshop on the Economics of Information Security*, 2004.

[24] N. Boris Margolin and Brian N. Levine. Informant: detecting sybils using incentives. In *Proc. of the 11th International Conference on Financial cryptography and 1st International conference on Usable Security*, FC '07/USEC '07, pages 192–207, Berlin, Heidelberg, 2007. Springer-Verlag.