In this paper, we investigate how secure Tor users are against end-to-end traffic correlation endpoints of the anonymity network. This attack is called Tor user and his destination by matching the traffic in both work [18]. This adversary is able to reveal the identity of the observer the traffic entering and leaving the anonymity network. To preserve anonymity in a situation where an adversary observes the traffic, relay adversaries. We show that our multiple-path strategies improve both client performance and anonymity at the cost of a server-side load increase. Finally, we discuss a possible architecture of Tor using Multipath-TCP, a TCP extension allowing multiple-path in the transport layer.

**Keywords**
Tor, anonymity, Multipath-TCP, Multiple-path Tor circuits, Security metrics, anonymity model, end-to-end traffic correlation, relay adversary

1. **INTRODUCTION**
Tor is an implementation of an Onion Routing protocol designed to provide anonymity over the Internet to TCP-based applications. Tor can be seen as a distributed overlay network run by volunteer-operated nodes where a Tor user is, for example, someone with incentives to keep privacy when surfing on the web. However, anonymity is a difficult property to guarantee and Tor is known by design to be unable to preserve anonymity in a situation where an adversary observes the traffic entering and leaving the anonymity network [18]. This adversary is able to reveal the identity of the Tor user and his destination by matching the traffic in both endpoints of the anonymity network. This attack is called end-to-end traffic correlation. In this paper, we investigate how secure Tor users are against this attack and we show the benefits of using multiple-path for Tor circuits. The paper is organized as follows: Section 2 recalls the necessary background about Tor and the end-to-end traffic correlation threat. Section 3 precises the security and anonymity models. We focus on end-to-end traffic correlation done by relay adversaries and we consider them perfect and instantaneous. Our work in section 4 focuses on designing a new path selection strategy in order to reduce the sensitivity to correlation attacks while improving the quality of communications thanks to multiple-path. We used and extended the Tor Path Simulator developed by Johnson et al. [10] in order to show the issue with the current path selection algorithm and the current state of the Tor network. This path simulator uses the real Tor network data (consensus and server descriptors) which are available in the CollecTor website [1]. In the second part of this section, we introduce the benefits of using multiple-path in Tor to improve the situation. In section 5, we use our security and anonymity models to construct some metrics which evaluate the anonymity and the performance of the Tor network when using the different algorithms introduced for path selection. Those metrics are a measure of anonymity only for the considered models and are not a measure of the real anonymity of the Tor network. As Syverson said [19], metrics might be fine as a measure of anonymity in an appropriate context. None is fine as the measure of anonymity. Therefore, the metrics we use is a heuristic to compare anonymity and performance of different path selection algorithms only valid for our models. Section 6 gives an analysis of the simulation results based on our metrics, showing that multiple-path in Tor could be a great opportunity in order to increase the performance and anonymity. Section 7 explains what we need to change in the Tor software to handle multiple-path, depending on the technology used in the transport layer.

2. **BACKGROUND**
Let us first remember how Tor is providing anonymity. The Tor network is composed of different types of nodes, onion proxies (Tor client), onion routers (relays), directory servers, bridges and hidden servers. If Alice wants to surf on the web anonymously, she needs an onion proxy also called a Tor client running on her computer. She needs to configure her browser to route its TCP flows towards the onion proxy, which can be seen as a gateway to the anonymity network. The onion proxy has the responsibility to provide the anonymity by creating a Tor circuit and assigning to it the TCP flows received from the browser. As you can see on Figure 1 this Tor circuit is typically composed of three onion.
3.1 Security Model

Performance and anonymity.

Indeed, if we had more nodes, we would experience more anonymity [20] and if we had more available bandwidth in the network, we would experience more throughput at mean. However, we will see in section 4 that the path selection algorithm plays a big role regarding performance and anonymity.

3. SECURITY AND ANONYMITY MODELS

3.1 Security Model

Since its initial design, Tor has been known to lack from efficient protection against end-to-end traffic correlation, as explained by Syverson et al. [20]. What we can do is making the end-to-end traffic correlation attack harder to achieve by offering more diversity in Tor nodes or in paths in the Internet.

In this model, an adversary controls a bunch of nodes in order to perform end-to-end traffic correlation when a Tor user is passing through one of its Guards and Exits nodes (Figure 2). We make the assumption that a correlation is instantaneous and perfect. The adversary is also able to control the properties of its nodes, such as the bandwidth. As we will see in the next section, the bandwidth could be a very accurate resource used by the adversary to improve the efficiency of its attack.

Notice that network adversaries are not taken into account in this model.

3.2 Anonymity Model

We will use the anonymity model from Claudia Diaz's Thesis [5] to construct anonymity metrics related to the end-to-end traffic correlation issue.

A general definition of anonymity comes from Pfitzmann and Hansen [16] and sees anonymity as the state of being not identifiable within a set of subjects, the anonymity set. The anonymity set stands for the set of all probable subjects. Those subjects are linked to anonymous actions. Those actions are called anonymous if an adversary cannot distinguish the subjects on which they are executed.

To model anonymity systems, we think in terms of Unlinkability. This term is also important for Diaz’s anonymity model. She borrowed a definition from Pfitzmann and Hansen:

Unlinkability of two or more items of interest (IOIs 1, e.g., subjects, messages, events, actions, ...) means that within the system (comprising these and possibly other items), from the attacker’s perspective, these items of interest are not more and no less related after his observation than they are related concerning his a-priori knowledge.

For Diaz’s anonymity model, anonymity may be defined as unlinkability between one or more IOIs. In our context, the IOIs are the packets passing through a Tor circuit. Because our security model states that a relay adversary successfully deanonymizes a Tor user once he owns a Guard and a Exit in a Tor circuit, we have unlinkability of packets when our relay adversary sees them at most once.

Therefore, the unlinkability of packets is dependent of the

---

1 IOIs means items of interest
path selection algorithm which chooses the appropriate nodes to be used for each Tor user’s circuit. The more the path selection algorithm provides diversity, the more the attacker will have difficulties to set up nodes seeing the packets at his Guard and Exit positions.

4. THE PATH SELECTION

The Tor path selection algorithm has known multiple changes since the beginning of its development. Those changes were brought to improve either client side performance, server side performance (relay), stability or anonymity. Most of them tried to improve the balance between performance and anonymity. Recently, Elahi et al [6] showed that changing Guard selection parameters could improve anonymity against end-to-end traffic correlation. The change made does not lower the performance on the client or the server side. Such improvement is really important, it is a full win change because performance is not impacted.

In path selection, this is the tricky question. How could we improve anonymity or performance without impacting the other property? Many papers tried to improve anonymity or performance but the other was always impacted.

The publication from Snader and Borisov [17] is a good example of strategies to either improve anonymity or performance but unfortunately decrease the other.

Many path selection algorithms have been suggested, Wacek et al. [21] published an empirical evaluation of them ([17, 2, 22]) and drew some interesting conclusions.

The bandwidth property is the main property to consider. Indeed, to be used, the Tor network must be usable. If we do not consider the bandwidth property, because of the state of the Tor network, we come up with a circuit having poor throughput performance, which is not enough for end users need. To face that issue, the Tor path selection algorithm uses a weighting strategy.

4.1 Current path selection algorithm

Tor weights the probability to select each node mainly according to the bandwidth property but also to the different status flags.

In order to see the consequences of the current weighted strategy, we have used the Tor path simulator developed by Johnson et al. to build a probability distribution for Guard and Exit positions. On Figure 3, you can see the impact of the bandwidth property in terms of cumulative probability to select the Exit node (y-axis) versus the nodes’ bandwidth sorted in decreasing order (x-axis).

Several important things can be concluded from this figure. First, regarding the data taken from 15th January 2015, we have a probability of 0.7 to choose among the top-100 Exit nodes (sorted by bandwidth) for any Tor circuit created. Secondly, the probability to choose among the top-400 is close to 1.0.

Thirdly, the evolution of the situation must be considered. We have about 815 Exits in the network around the 15th February 2013, for a total of about 3100 onion routers (≈ 26%). We have about 920 Exit onion routers on 16th December 2013, for a total of about 4800 onion routers (≈ 19%). And about 1130 on 15th January 2015 for a total of about 7000 onion routers (≈ 16%). As you can see, even if the number of Exit onion routers increases through time, the situation remains bad in Figure 3 from a diversity viewpoint in the Exit selection. Therefore, saying that we need more nodes in the Tor network is not really true. What we need is more big nodes which will get enough weight in the selection process to change the situation.

Currently, many Exit nodes with small bandwidth capacity are available and, fortunately from a performance viewpoint, they are almost never selected. With a single path strategy, we have no choice, we need this unfair weighting selection. The idea of using multiple-path in Tor solves this issue. Instead of using only one Exit node, we cumulate the bandwidth of several Exit nodes. Consequently, it becomes possible to use the available diversity in the network, hence enhancing the anonymity.

Since we are interested in end-to-end traffic correlation by relay adversaries, let us see also what is the situation with Guard selection. Figure 4 shows, as for the Exit selection, the cumulative probability to select Guard nodes (y-axis) when they are sorted by bandwidth decreasing (x-axis). The situation is not as bad as for Exit selection because Guard nodes are good nodes, with enough bandwidth to be called fast and enough uptime to be called stable. Hence, the path selection algorithm weights nodes less disproportionately. Notice that the number of available guards is quite
To summarize the issue, we could say that the state of the Tor network has led the developers to play with weighted selection mainly over the bandwidth property of nodes. This choice was done to make the Tor network usable w.r.t service’s needs in bandwidth, such as browsing heavy websites. Unfortunately, this is a severe drawback against end-to-end traffic correlation.

4.2 Using a multiple-path Tor circuit

Regarding the issue of the current path selection algorithm which is caused by the state of the Tor network, changing the Tor circuit in order to use multiple-path solves the problem. At this point, it could be a little confusing. The question to focus on is the following: how can a new path selection algorithm for multiple-path Tor circuits change the state of the Tor network?

If we want to change the state of the Tor network, which means providing more bigger nodes, we need a lot of individuals putting fund together to run big nodes. With a multiple-path Tor circuit, we can put this idea of collective collaboration in practice to get bigger nodes by simply using more than one circuit at a time. A multiple-path Tor circuit like Figure 5 allows the Tor user to cumulate the bandwidth of multiple Guards, Middles and Exits nodes. Hence, the small nodes of the Tor network become usable and useful since they can be cumulated with others. Moreover, a multiple-path Tor circuit with a multiple-path protocol on the transport layer provides so much stability that we could handle unstable nodes easily and say goodbye to the problem of potentially loosing the connection with the service we are using.

To summarize, the goal of the new path selection algorithm for multiple-path Tor circuit is to achieve the following points:

- Performance. The bandwidth provided to the Tor users by the new path selection algorithm must be equal or better than the bandwidth provided to the current path selection algorithm, at average.
- Security. The probability to get compromised nodes must be lower than the probability with the current path selection algorithm.
- Stability. If a node crashes in the circuit, the other nodes handle the stream and the connection on the transport layer is not closed.

In the context of a multiple-path Tor circuit, the "weight" of each node must also depend on the current state of the Tor circuit. For example, if we already have a circuit established with 3 running Exits and a correct bandwidth, a forth one could be selected uniformly at random. Hence, a weighting strategy may not be appropriate anymore.

We have designed and implemented several strategies inside the Tor path simulator in order to experience the balance between performance and anonymity.

4.2.1 Selecting Guards

We start by choosing the appropriate Guards for our circuits. The entry Guard design is applied as Tor does today: we select a number $G$ of Guards from the pool forming the set $S_G$. The Tor proxy only uses the Guards contained in $S_G$ as entry point for a long time (between two and three months currently). This number $G$ of Guards is dependent of the strategy we use to balance the anonymity and performance.

A first strategy, called $EntryGuards N$, is used to balance towards anonymity by uniformly selecting at random these $G$ guards in order to use $N$ of them at the same time in the multiple-path circuit with relation $G = 3 \times N$. If $N = 1$, we have the same set of Guards as Tor does today but with $N = 1$, the performance in bandwidth is not sufficient at mean because we choose the Guards uniformly at random. However, using $N = 2$ or $N = 3$, the bandwidth could become sufficient even with a uniform selection thanks to the good properties of Guard nodes. This claim will be verified in the Analysis part of this paper.

A second strategy, called $BandwidthRate M$ K/M/s, is used to balance towards performance by using enough Guards to meet the requirement of $M$ K/M/s by cumulating their bandwidths. We select $G$ Guards and we construct the set $S_G$ with the constraint $Bandwidth(S_G) \geq 3 \times M$. The tricky question is the way in which we weight the selection of Guards. We need a trade-off between a selection as large as possible and a reasonable size of the set $S_G$ to minimize the chance to select a Guard node owned by an adversary. The Guards are selected with the help of the following function:

$$f_s(x) = 1 - \frac{2^{s+2}}{2^s}$$

with $x$ random in $[0, 1]$ and $s$ the size of $S_G$. The selection weights more the bigger2 Guards by applying the following rules: each time we select a Guard to add it in $S_G$, we compute $T = sizeof(all\_guards\_list) \times f_s(x)$ with a new $x$ for each selection. Notice that $f_s(x)$ computes a random value

2bigger means more bandwidth
between $[0,1]$ biased by the value of $s$. Then, we construct the list $L = sublist(all\_guards\_sorted\_list, T)$ which is the sublist of the first $T$ elements from the list of all guards sorted by decreasing bandwidth. Finally, we pick uniformly at random a node from the sublist $L$. The use of this function is inspired from Snader and Borisov [17] but here, we seek to have a dynamic trade-off between anonymity and performance. $f_s(x)$ can be seen as a way to easily weight the selection of Guards towards bigger bandwidth and the weight is re-evaluated at each selection. We select more uniformly the last Guards to be added to $S_G$ than the first, which provide at mean a fewer number of Guards for $S_G$ than if they were all selected uniformly. This is our trade-off between a selection as large as possible and a reasonable size of the set $S_G$.

4.2.2 Selecting and rotating Exits

The selection of Guards has set the bandwidth of our multiple-path circuit, therefore we try to select enough Exit nodes to get a cumulative bandwidth equal or greater than the cumulative bandwidth of the Guards. We use the same function $f_s(x)$ to weight towards performance but $s$ has a different meaning. For Exit selection, $s$ is the number of Exit nodes being currently used in the multiple-path circuit. The selection process starts by building an ordered list of Exit nodes sorted by decreasing bandwidth, then we compute $T = sublist(all\_guards\_sorted\_list, T)$. Finally, we uniformly pick one of the nodes inside the list $L = sublist(exit\_nodes\_sorted\_list, T)$ at random.

The circuit change through time like the current Tor design for which the circuit is simply changed every 10 minutes by default. For our multiple-path Tor circuit, every 10 minutes by default, it changes some of its Tor circuits. In fact, we change some of the Exit nodes every 10 minutes and we kill every subpaths (Tor circuit) passing through those Exit nodes. The number of Exit nodes dropped depends on the number of Guards used and their bandwidth.

$$\sum \frac{\text{Bandwidth}(\text{guardsInUse})}{\#\text{guardsInUse}}$$

Then, we drop enough Exits with cumulated bandwidth that match the above computation.

4.2.3 Selecting Middles

We consider only the set of unflagged Guard or Exit relays and we can select unstable non-valid Middle nodes but we always have at least one stable Middle node in the multiple-path circuit. The selection is done with the weighting strategy as Tor does today.

Also, we do not rotate middle nodes unless:

- The node is down or hibernating
- All subpaths through the middle node have been removed by the Exit rotation algorithm

5. METRICS

Measuring anonymity is a tough problem. Anonymity on the Tor network, from our opinion, depends on multiple factors. First, it depends on the items of interest which is in our case the packets through the circuit (and not outside). Secondly, it depends on the kind of adversary considered.

Once you have considered your kind of adversary, you can consider building some metrics that can be applied for the studied situation. Hence, the following introduced metrics will not be used to measure the anonymity of the Tor network, which is probably too complex, but our metrics will give an idea of how anonymous we are w.r.t. the anonymity model and the adversary.

5.1 Anonymity Metrics

5.1.1 Shannon entropy

This metric will be used to quantify the quantity of information given to the adversary about Guard selection and Exit selection

$$H(X) = - \sum_{i=1}^{N} p_i \log_2(p_i)$$

$X$ is a discrete random variable with probability mass function $p_i = Pr(X = i)$, which is in our context the probability to select the node $i$. The more the path selection algorithm gives information to the adversary, the smaller is the value of this metric. Or, if you prefer, the quantity of information that the adversary obtains by observing the system is $H_M - H(X)$ with $H_M$ the maximum entropy of the system (from an uniform distribution).

5.1.2 Degree of uniformity

The degree of uniformity has been defined by Diaz in her Ph.D thesis [5] as the degree of anonymity. We change the name because it has a different meaning in our context.

$$\text{degree of uniformity} : \frac{H(X)}{H_M}$$

This is a normalized version of the Shannon entropy, which is useful to compare sets that do not have the same number of elements. We have chosen to use this metric in order to be able to compare the uniformity in Exit selection versus the Guard selection without being impacted by the different number of nodes for each set. Also, this metric can be used to compare the Guard or Exit selection for different dates.

5.1.3 Degree of uniformity of circuit selection

This metrics and the following (Guessing entropy) have been chosen to quantify the likelihood to be subjected to an end-to-end traffic correlation. We have:

$$H(Y) = - \sum_{i=1}^{N} \sum_{j=1}^{K} p_{i,j} \log_2(p_{i,j})$$

$p_{i,j}$ being the probability to have the Guard $i$ with Exit $j$ in the circuit.

$$\text{Degree of uniformity of circuit selection} : \frac{H(Y)}{H_M}$$

$H_M$ refers to the maximum entropy, which is computed by $log_2(NK)$. Notice that a degree of 1 is impossible to obtain.
for the current path selection algorithm in Tor because of the different additional constraints (e.g., we can’t have two same /16 addresses in the circuit, the Guard and Exit cannot be the same node, ...).

The degree of uniformity of circuit selection is a metric that quantify the uniformity for the available pairs Guard-Exit in the Tor network given by the path selection algorithm, ignoring the middle nodes that have no influence on the end-to-end traffic correlation attack w.r.t our security and anonymity models.

5.1.4 Guessing Entropy
This metric is an answer to the following question: How many nodes must a relay adversary compromise to succeed with an end-to-end traffic correlation at mean ?

\[
\sum_{i=1}^{N} p_i \cdot \frac{1}{p_i}
\]

In the context of the path selection algorithm, \( N \) is the total number of nodes (Guards + Exits) involved in the selection process and \( p \) is a vector of probability of size \( N \) with:

- \( p_i \) the probability to win with the addition of an i-th node, which is the probability that the relay adversary wins an end-to-end traffic correlation thanks to the i-th node (\( p_1 = 0 \))

This metric provides the average number of nodes to compromise to be on end points of any Tor circuit created by any Tor users. This metrics takes into account both the quality of the probability distribution and the number of nodes. It means that we have a metrics to evaluate the improvement on anonymity of the Tor network through time and for different path selection algorithms.

5.2 Performance Metrics
5.2.1 Bandwidth
This metric is the average bandwidth the Tor users get over the considered time period. It is a client-side performance metrics.

5.2.2 Number of key-exchange protocols run per hour
This is a server-side performance metrics. The more there are construction of circuits, the heavier is the CPU-load on the servers.

A performance metrics on server-side matters in order to see if a path selection algorithm is reasonable, especially with multiple-path for which we get many more circuit construction requests.

6. RESULT ANALYSIS
Our metrics have been used on the probability distributions built with the Tor path simulator for Tor 0.2.4.x and for our multiple-path strategies. The simulations use the real Tor data making a full size simulation with all nodes that were available at the date studied. Those simulations are simple. Every 10 minutes the user model sends a connect stream request to an http website. In the case of Tor 0.2.4.x, because it is configured to create a new circuit every 10 minutes, the stream was handled in the simulator each time by a new circuit. Consequently, the number of circuits created for a one day period is \( N \times 144 \) with \( N \) the number of simulated clients. We have used \( N = 20000 \) for Tor 0.2.4.x, hence creating a total of 2880000 circuits. Regarding our multiple-path strategies, it is a little more tricky. We have many Tor circuits inside our multiple-path circuit (See Figure 5 for recall) and a part of them are renewed every 10 minutes, as explained in section 4.2.2. Hence, the number of Tor circuits is variable. We have used \( N = 10000 \) for our multiple-path strategies, which provides a sufficient number of created circuits for each strategy in order to make the probability distribution relevant.

The goal of this section is the analysis of anonymity and performance of Tor 0.2.4.x and our multiple-path strategies. The simulation over different dates provides some intuition about the evolution of the Tor network regarding anonymity. Table 1 shows the different strategies we consider for multiple-path circuits.

The first strategy balances towards anonymity using two Guards uniformly and randomly selected. The maximum number of Exit nodes is set to 5 and we always have a full mesh between nodes to improve stability and maximize the throughput (e.g. for Figure 5, the fullmesh is used to obtain 27 Tor circuits between source and destination).

The second strategy is used to balance towards performance, each user obtains a multiple-path circuit with a bandwidth at least of 50 000 K/\( s \) if the maximum number of Guards and Exits used in the circuit is not higher than 5. The minsubpaths value means that the path manager tries to minimize the number of Tor circuits used in the multiple-path circuit, while trying to maximize the throughput.

Finally, the third strategy decreases the stability of the first one to balance towards server performance.

Therefore, we have chosen three strategies, one to balance towards anonymity and stability, one to balance towards performance and one to balance towards anonymity and server performance.

Table 2 shows our results. Red metrics means that lower results are better and black metrics means higher results are better. Moreover, only numbers with the same special text form are comparable to each other for the different dates.

6.1 Tor 0.2.4.x
Let us first consider the results for Tor 0.2.4.x. The different anonymity metrics provide an empirical evaluation of the anonymity of the Tor network against relay adversaries based on path selection simulations. We can confirm our first intuitions introduced in Section 4. We have a better uniformity from Guard selection than the uniformity from Exit selection as given by the degree of uniformity. Consequently, it is more costly for a relay adversary to maintain the same level of threat in Guard selection than with Exit selection. Moreover, the Guessing Entropy gives an idea of the number of nodes the adversary should compromise at average to deanonymize any Tor user for each circuit built (a new circuit is built every 10 minutes). Regarding the 15th Feb 2013, 198 nodes is a really small value. Hopefully,
during the last couple of years, the Tor network has become better with more good nodes (giving a sufficient amount of bandwidth). This is true for Guard nodes, but for Exit the situation has not really changed between the 16th Dec 2013 and the 15th Jan 2015 as indicated by the degree of uniformity for Exit selection. Even if we have more Exit nodes, the probability distribution of selection remains really bad. Regarding performance metrics, it is not really interesting to compare the bandwidth between dates since it could vary a lot from a consensus to the next one. However, we can compare the bandwidth we got from our different strategies for the same date.

The Tor network has to handle 3 key-exchange protocols per circuit construction, thus the amount of the last metrics is directly dependent of the number of circuit built per Tor user. We have simulated 10 000 Tor users for each strategy and regarding Tor 0.2.4.x, a simulation for 10 000 Tor users leads to 180 000 key-exchange protocols run per hour (3 key-exchange protocols run every 10 minutes). We do not take into account the clean circuits created and we have not simulated the possible issues arising from bad behaviours of Tor node operators, like intentionally killing the circuit of some Tor users. Consequently, this number is always underestimated. We consider this value as the load on the onion routers at average.

### 6.2 Comparison with the multiple-path strategies

A first observation on the results shows that all multiple-path strategies improve the situation regarding anonymity, all metrics give better results, even for the multiple-path strategy that balances towards performance (strategy 2). Before having a closer look on anonymity metrics, we need to know if the performances are sufficient enough. Regarding the bandwidth, we got higher bandwidth than Tor 0.2.4.x with multiple-path strategies balanced towards anonymity for both the 16th Dec 2013 and 15 Jan 2015 but not for the first studied date. The reason is simply because in February 2013, the total amount of Guard nodes’ bandwidth was a third of the current total amount of bandwidth, hence using two uniformly and randomly selected Guards is not enough on average to have more than the bandwidth obtained by Tor 0.2.4.x with its weighted selection.

Regarding strategies 1 & 3, we select uniformly and randomly the Guards, hence the degree of uniformity for Guard selection should theoretically be 1.0. for each date. Regarding our results, this metrics gives a value close to 1.0 as expected. Globally, we have an improvement on anonymity and client-side performance at the cost of the server-side performance (on onions routers). The nodes in the Tor network must handle more circuit creation requests. How many more is reasonable? We know that doubling the amount of circuit creation requests on average could lead to denial of service of some nodes [13]. However that issue happened when the TAP handshake was the default protocol to establish circuit. Today, with the stable version of Tor 0.2.4.x, the protocol has changed to the NTor protocol [8] which is less cpu-expensive. Knowing how much more load the Tor network could support could be interesting. However we are almost sure that two times the current load on average is reasonable, giving the fact that the NTor protocol is used and that multiple-path strategies spread the load on a bigger part of the Tor network.

Moreover, if we want to reduce the load on onion routers, we could change the default 10 minutes interval between circuit updates. Simply moving to 20 minutes would let the Strategy 2 be reasonable, since the number of key-exchange protocols run per hour would be divided by 2. Anyway with the current default values in the Tor network, our strategy 1 is not reasonable, the full mesh strategy brings a strong stability but induces too much load on server-side. Our strategy 2 simply creates too many Tor circuits at mean to cover the 50Mbits bandwidth requested. A smaller value could lead to a reasonable result. Our Strategy 3 could be a good candidate since the additional load is reasonable. Regarding the 15th January 2015, the most recent date, this strategy brings 2.33 times the bandwidth on average to the Tor users. The improvement of performance was not really targeted, it is just a consequence of using multiple-path Tor circuits with aggregation of bandwidth. What really matters is the fact that the Tor network could be a lot more secure with multiple-path w.r.t relay adversaries.

### 7. MOVING TO MULTIPLE-PATH ?

Multiple-path has a huge potential to increase performance and anonymity of the Tor network. Previously, there have been several papers promoting different kinds of multiple-path for different purposes. AlSabah et al [3] introduce Conflux, a traffic-splitting approach between the onion proxy and an Exit node. They have noticed improvement over throughput and latency. Karaogly et al. [11] demonstrate the potential of multiple-path to improve congestion avoidance using Multipath-TCP [4], a new TCP extension allowing more than one path on the transport layer.

With our work, we have also shown that multiple-path would increase anonymity by using much better the diversity of the Tor network in the selection process during the construction of circuits.

Therefore, could we move towards a multiple-path solution? Some difficulties arise. First, we need a new protocol on the transport layer which aggregates the bandwidth from the different Tor circuits contained in the multiple-path circuit. A good candidate is Multipath-TCP (MPTCP for short) for which an implementation in the linux kernel exists [15]. Other protocols could fit but they are out of the scope of this paper.

If we want to use MPTCP (or another multiple-path protocol), it will cause many changes to the original Tor software.

---

### Table 1: Multiple-path strategies

<table>
<thead>
<tr>
<th>label</th>
<th>Guard option</th>
<th>pathManager</th>
<th>Maximum #Guards and #Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategy 1</td>
<td>EntryGuards 2</td>
<td>Fullmesh</td>
<td>2 Guards, 5 Exits</td>
</tr>
<tr>
<td>Strategy 2</td>
<td>BandwidthRate 50000</td>
<td>minsubpaths</td>
<td>5 Guards, 5 Exits</td>
</tr>
<tr>
<td>Strategy 3</td>
<td>EntryGuards 2</td>
<td>minsubpaths</td>
<td>2 Guards, 5 Exits</td>
</tr>
<tr>
<td>Date</td>
<td>Metrics</td>
<td>Tor 0.2.4.x</td>
<td>Strategy 1</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------</td>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>15th Feb 2013</td>
<td><strong>Quantity of info. leaked for Guard selection [bits]</strong></td>
<td>1.5352</td>
<td>0.0857</td>
</tr>
<tr>
<td></td>
<td><strong>Quantity of info. leaked for Exit selection [bits]</strong></td>
<td>2.9125</td>
<td>2.7341</td>
</tr>
<tr>
<td></td>
<td><strong>Degree of uniformity of Guard selection</strong></td>
<td>0.8429</td>
<td>0.9913</td>
</tr>
<tr>
<td></td>
<td><strong>Degree of uniformity of Exit selection</strong></td>
<td>0.6992</td>
<td>0.7262</td>
</tr>
<tr>
<td></td>
<td><strong>Degree of uniformity of circuit selection</strong></td>
<td>0.7611</td>
<td>0.8321</td>
</tr>
<tr>
<td></td>
<td><strong>Number of key-exchange protocol run per hour</strong></td>
<td>198</td>
<td>484</td>
</tr>
<tr>
<td></td>
<td><strong>Bandwidth at mean [Kb/s]</strong></td>
<td>10 924</td>
<td>× 0.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180 000</td>
<td>× 3.69</td>
</tr>
<tr>
<td>16th Dec 2013</td>
<td><strong>Quantity of info. leaked for Guard selection [bits]</strong></td>
<td>1.6051</td>
<td>0.1151</td>
</tr>
<tr>
<td></td>
<td><strong>Quantity of info. leaked for Exit selection [bits]</strong></td>
<td>2.3831</td>
<td>1.8915</td>
</tr>
<tr>
<td></td>
<td><strong>Degree of uniformity of Guard selection</strong></td>
<td>0.8503</td>
<td>0.9893</td>
</tr>
<tr>
<td></td>
<td><strong>Degree of uniformity of Exit selection</strong></td>
<td>0.7585</td>
<td>0.8131</td>
</tr>
<tr>
<td></td>
<td><strong>Degree of uniformity of circuit selection</strong></td>
<td>0.8021</td>
<td>0.8752</td>
</tr>
<tr>
<td></td>
<td><strong>Guessing entropy [nodes]</strong></td>
<td>334</td>
<td>784</td>
</tr>
<tr>
<td></td>
<td><strong>Bandwidth at mean [Kb/s]</strong></td>
<td>13 285</td>
<td>× 1.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180 000</td>
<td>× 4.86</td>
</tr>
<tr>
<td>15th Jan 2015</td>
<td><strong>Quantity of info. leaked for Guard selection [bits]</strong></td>
<td>0.8678</td>
<td>0.1204</td>
</tr>
<tr>
<td></td>
<td><strong>Quantity of info. leaked for Exit selection [bits]</strong></td>
<td>2.3559</td>
<td>1.4729</td>
</tr>
<tr>
<td></td>
<td><strong>Degree of uniformity of Guard selection</strong></td>
<td>0.9148</td>
<td>0.9886</td>
</tr>
<tr>
<td></td>
<td><strong>Degree of uniformity of Exit selection</strong></td>
<td>0.7656</td>
<td>0.8559</td>
</tr>
<tr>
<td></td>
<td><strong>Degree of uniformity of circuit selection</strong></td>
<td>0.8327</td>
<td>0.8998</td>
</tr>
<tr>
<td></td>
<td><strong>Guessing entropy [nodes]</strong></td>
<td>494</td>
<td>764</td>
</tr>
<tr>
<td></td>
<td><strong>Bandwidth at mean [Kb/s]</strong></td>
<td>10 893</td>
<td>× 2.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180 000</td>
<td>× 7.74</td>
</tr>
</tbody>
</table>

See Figure 6 for an example of a multiple-path architecture for Tor using MPTCP. Representing a multiple-path Tor circuit with a layered architecture helps focusing on the principals difference with the original Tor software. First, the initiator (Tor proxy) must handle multiple-path circuits with all related needs such as new path selection algorithms, congestion avoidance, new constraints, ... Moreover, we need different TLS channels for all TCP subflows of the MPTCP connection.

Regarding the intermediate nodes (Guard and Middle), they do not need any changes since they are forwarding data. The Exit node is responsible to initiate the connection with the destination. In the current design of Tor, the Exit node waits the information IP:port to be received from the Tor proxy through the circuit and initiates its TCP connection with the destination once the information is received. Here, we need to establish an MPTCP connection with the server. This is more difficult because the connection must be bound between the Tor proxy and the destination and initiated by the Exit node. Therefore, we cannot use current Exit nodes to establish multiple-path circuit.

Another problem is that, even if MPTCP is a standardized protocol (RFC 6824 [7]), it has not been integrated in the Linux kernel yet or in any other kernel by default. Hence, MPTCP-capable servers running services such as websites only exist currently in the scientific community for research purposes. However, motivations to use multiple-path on the transport layer are widespread and today it is already used by Apple since iOS 7.0. They have integrated MPTCP in their kernel to create a backup channel in the cellular network when Siri is used. The deployment of multiple-path capabilities on the transport layer with MPTCP is a question of time.

8. CONCLUSION
One of the major problems with Tor today is its unfair path selection algorithm. The path selection algorithm has to be unfair and promote larger nodes because Tor has to be usable. Indeed, without the weighted selection, Tor users would select most of the time nodes that cannot meet their requirements, even for web browsing. Consequently, the anonymity provided by Tor is not difficult to bypass with a bunch of compromised nodes, as demonstrated Johnson et al. [10]. Moving towards multiple-path Tor circuits which aggregate the bandwidth from different Tor circuits could be a solution. We have implemented our path selection algorithm and tested different strategies on the real Tor data provided by the archives of consensus and server descriptors. We have also provided some anonymity metrics relevant for
the context of node selection and relevant to quantify the anonymity against relay adversaries. Our results confirm that Tor is not as secure as previously thought. Moreover, we have found an acceptable trade-off anonymity/performance for multiple-path strategies which gives much better anonymity and bandwidth to Tor users. Finally, we specify that a multiple-path Tor network that aggregates bandwidth from multiple Tor circuits is only possible with a new protocol on the transport layer.

9. REFERENCES