J Supercomput
DOI 10.1007/s11227-012-0788-4

How to block Tor's hidden bridges: detecting methods and countermeasures

Ming Yang · Junzhou Luo · Lu Zhang · Xiaogang Wang · Xinwen Fu

Abstract Tor network has been widely used for protecting the privacy of users while accessing various online services. Since Tor can be easily blocked by blacklisting the publicly published Tor relays, the hidden bridges-based blocking-resistance mechanism is designed and implemented in the current Tor network. Any user can subscribe a tuple of three bridges via email, https, twitter etc. However, we have found that there exist high correlations among those published tuples, which can be exploited to effectively detect hidden bridges by monitoring the outbound traffic from a controlled network. When Tor clients try to connect chosen hidden bridges, multiple SYN packets with consecutive source ports will be sent almost simultaneously, destining for different hosts. If any destination IP contained among such packets belongs to a known bridge, all others can then be inferred to be of bridges too. By recording and analyzing a series of traffic segments satisfying the above packet features, the hidden bridges used in a controlled network can be detected and further blocked. According to different available computing and storage resources, we proposed both online and offline detecting methods. Both analytical and simulation results verify the high correlation among published bridge tuples, validating the feasibility of our methods. By configuring optimized detecting parameters in the real-world experiments, we can achieve a detection rate of 86.7 % with a 0.85 % false-positive rate for online detection, and a 98.4 % detection rate with a 0.62 % false-positive rate for offline detection. To make up the flaws in Tor's current blocking-resistance mechanism, we also provide some countermeasures from the perspective of Tor network and users, respectively.

M. Yang (✉) · J. Luo · L. Zhang · X. Wang
School of Computer Science and Engineering, Southeast University, Nanjing 211189, P.R. China
e-mail: yangming2002@seu.edu.cn

X. Wang
Changzhou College of Information Technology, Changzhou 213164, P.R. China

X. Fu
University of Massachusetts, Lowell, MA 01854, USA

Published online: 09 June 2012

Keywords Privacy · Tor network · Block resistance · Bridge

1 Introduction

Anonymous communication systems such as Tor [1] have been studied extensively since David Chaum introduced the mix in 1981 [2]. As one of the most popular anonymous networks, Tor employs widely deployed onion relays to anonymize TCP traffic for protecting the privacy of users. However, the Tor network can be easily blocked if an adversary controls Tor user's connection to the Tor network by blacklisting all the publicly published relay IP addresses. In order to address such a problem, the hidden bridges-based blocking-resistance mechanism is designed and implemented in the current Tor network [3].

In the Tor network, hidden bridges are a set of secret entry nodes, which can be used by users to establish secure connections to the main Tor network and directory authorities, thereby building circuits and connecting to the rest of the Internet privately and securely. There are several ways for bridges to be published, such as email, https and twitter etc. To avoid being enumerated, Tor bridges are published with a smart publishing strategy. In response to user's subscription, only partial bridges in bridge pools are published in a certain period, in the form of bridge tuple that contains three bridges each. Therefore, it is hard to enumerate all the bridges in limited time due to the publishing strategy. However, Tor may be blocked without all the bridges being enumerated. In this paper, we only focus on the detection of bridges that are used in a controlled network.

Our detecting methods rely on two observations. (1) Since only partial bridges are published in a certain period, the same bridges will occur repeatedly, resulting in high correlation among tuples. (2) When Tor clients try to connect chosen hidden bridges, multiple SYN packets with consecutive source ports will be sent almost simultaneously, destining for different hosts. If any destination IP contained among such packets belongs to our current known bridge set, all others can then be inferred to be of bridges too. Therefore, under the assumption that we have got an initial bridge set, the bridges used in a controlled network can be effectively detected by monitoring the traffic pattern and correlating suspected bridge tuples with known bridges. According to different available computing and storage resources, we proposed both online and offline detecting methods. The former can discover potential bridges in real time with lower detection rates, while the latter can achieve higher detection rates with more storage resources required.

Our main work includes the following: (1) We found and verified the high correlation among published bridge tuples, by extensive simulation on practical dataset obtained via email and https. (2) We proposed online and offline detecting methods based on the bridge traffic pattern by monitoring the outbound traffic from the controlled network. (3) We built probabilistic models of online and offline detection, respectively, and conducted Monte Carlo simulation to validate the effectiveness of our methods. (4) We deployed our methods in our campus network and conducted real-world experiments to evaluate the detection rates and false-positive rates of the proposed detecting methods. The empirical results demonstrate that the hidden Tor bridges used in a large controlled network can be effectively and efficiently discovered by our methods, thereby successfully blocking the access to Tor network. In order to mitigate our blocking methods, we also provide some countermeasures for securing Tor's bridge design.

The remainder of this paper is organized as follows: We describe the background in Sect. 2. The detailed Tor bridge detecting methods along with some discussions are presented in Sect. 3. We analyze the effectiveness of our methods by establishing statistical models in Sect. 4. Our experimental results are presented in Sect. 5, validating our findings. Some countermeasures against the blocking methods are discussed in Sect. 6. We review previous work in Sect. 7. We conclude the paper along with some future research directions in Sect. 8.

2 Background

In this section, we introduce Tor network and discuss current Tor bridge mechanism, including its architecture and publishing strategy.

2.1 Tor network

Tor is a popular overlay network for anonymous communication on the Internet, which is an open source project providing anonymity service for TCP applications. It exploits onion routing that is based on layered encryption to hide the communication relationship of TCP traffic. Figure 1 shows the fundamental architecture of the Tor network consisting of four basic entities.

(1) Client. The client runs local software called onion proxy (OP) to construct circuits and anonymize the client data into the Tor network.
(2) Server. The server runs various TCP applications such as a web service and anonymously communicates with clients.
(3) Onion routers (OR). Onion routers are special proxies that relay the application data between clients and servers. Each OR maintains a Transport Layer Security (TLS) connection to other ORs or OPs. A circuit is a path of three onion routers by default through the Tor network, namely entry onion router, middle onion router and exit onion router, respectively.

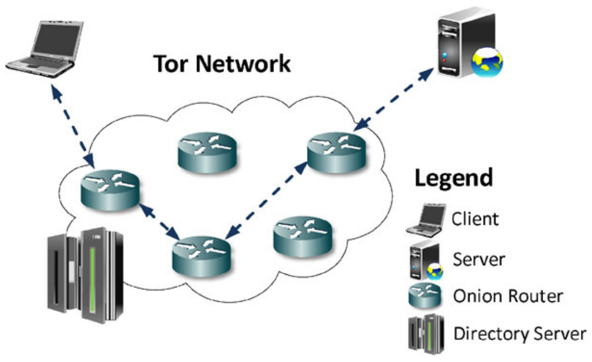

Fig. 1 Tor network

(4) Directory servers. They are responsible for maintaining and providing onion router information for Tor clients.

Functions of onion proxy, onion router and directory server are all integrated into a single Tor software package. A client can edit a Tor's configuration file and configure a computer to have any combination of those functions.

To send TCP data through a circuit, a client first chooses three onion routers and then constructs a Tor circuit. The onion router information can be downloaded from directory servers in the Tor network. The entry router knows the client's identity, i.e., the client's IP address. The exit router knows both the destination's identity and the actual TCP data sent, and the middle router simply exchanges encrypted data between the entry router and the exit router along a particular circuit. The application data are packed into equal-sized cells and encrypted in the onion-like fashion. All cells are successively encrypted with three session keys that have been negotiated by OP with each OR in the path. Therefore, anyone monitoring the encrypted traffic cannot use cell size to help identify a client.

2.2 Tor bridges

Since Tor routers operate on a few thousands of distinct IP addresses available publicly, an adversary could enumerate and block them all with little trouble. To provide a means of ingress to the network, Tor introduces a hidden set of entry nodes, i.e. bridges, which an adversary will not be able to enumerate. In Tor network, a bridge is a client node that volunteers to help censored users access Tor by serving as an unlisted, first-hop relay.

Figure 2 shows the Tor bridge architecture. Bridge authorities aggregate and track bridges. Bridges periodically publish relay descriptors only to the bridge directory authorities, which include summaries of their keys, locations, bandwidths, etc., signed by their long-term identity keys. The bridge authorities only give out a relay descriptor if you already know its identity key. That is, any user can keep up-to-date on a bridge's location and other information when and only when he knows about it, but cannot enumerate the bridges. The identity key, IP address, and directory port for each bridge authority ship by default with the Tor software, so the bridges can be confident that they are published to the right location, and the blocked users can establish an encrypted authenticated channel. If a blocked user knows the identity keys of a set of bridges, and he has correct address information for at least one of them,

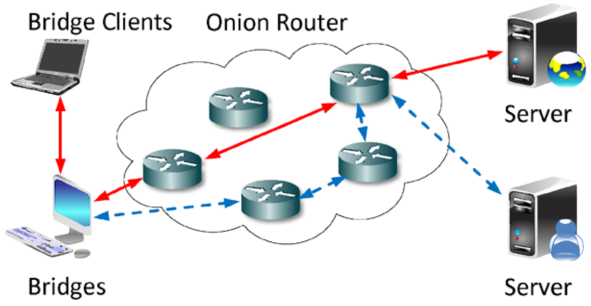

Fig. 2 Tor bridge architecture

he can use that one to make a secure connection to the bridge authority and update his knowledge about other bridges. He can also use it to make secure connections to the main Tor network and directory authorities, so he can build circuits and connect to the rest of the Internet. All of these updates happen in the background: from the blocked user's perspective, he just accesses the Internet via his Tor client like always.

Bridges use Tor to publish their descriptors privately and securely, so even an adversary monitoring the bridge directory authority's network cannot make a list of all the addresses contacting the authority. Bridges may publish to only a subset of the authorities, to limit the potential impact of an authority compromise. Any Tor client can subscribe a bridge tuple consisting of three published bridges through any of 6 means, including email, https, twitter etc. The current Tor network adopts a smart bridge publishing strategy to defend against enumerating all bridges, where only partial bridges from bridge pool can be permutated and published in a certain period of subscription by email or any other means. Although it is hard to enumerate all potential bridges in limited time by a particular subscription method, there exists a high correlation among published bridge tuples, which is the basis of our detecting methods.

3 Tor bridge detecting methods

In this section, we first introduce the basic principle of our detecting methods. We then present both online and offline detecting algorithms followed by discussions.

3.1 Basic idea

We assume that an adversary controls Tor users' connections to the Internet, and tries to detect all bridges used in the controlled network. Our detecting methods reside in the observation: When connecting to Tor network via bridges, the client first creates a series of non-blocking sockets, and then connect those sockets one after another in a short period for establishing TCP connections to chosen bridges. The chosen bridges include two parts: bridges configured by users and bridges cached by Tor software. Like most applications, the Tor client does not bind pre-determined source ports for those connections. Conversely, it is the operating system that assigns ephemeral source ports for connections [4] to chosen bridges. Such ephemeral source ports are usually consecutively allocated in widely used operating systems such as Windows platforms. As a result, multiple SYN packets with consecutive source ports will be sent almost simultaneously, destined for different bridges. We herein present two definitions followed with our observations.

Definition 3.1 An *effective traffic segment (ETS)* refers to the traffic data containing a series of SYN packets with consecutive source ports. Such SYN packets are all originated from the same source IP and destined for different destination IP addresses within a short time slot.

As shown in Fig. 3, the parameter T means the maximum duration time for ETS, which is determined by the packet transmission interval and the network jitter between the source host and the detecting point. Accordingly, we specify T as the size

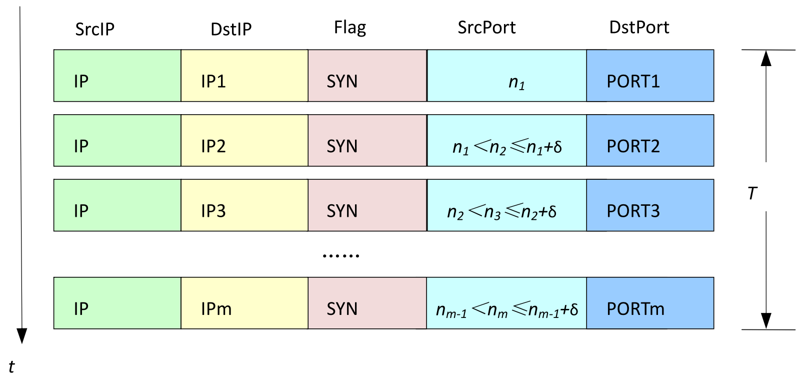
SrcIP
DstIP
Flag
SrcPort
DstPort
IP
IP1
SYN
n_1
PORT1
IP
IP2
SYN
$n_1 < n_2 \leqslant n_1+\delta$
PORT2
IP
IP3
SYN
$n_2 < n_3 \leqslant n_2+\delta$
PORT3
......
IP
IPm
SYN
$n_{m-1} < n_m \leqslant n_{m-1}+\delta$
PORTm
T
t

Fig. 3 Effective traffic segment

of the sliding time window used for traffic caching in our detecting algorithms. n_1 can be any possible port chosen by the operating system then. δ means the appropriate port interval between so-called consecutive SYN packets, whose value will be discussed later. The requirement for different destination IPs resides in the fact that Tor clients will not access the same bridge more than two times within T, where "two times" occurs only when a bridge is configured and cached at the same time. Therefore, some common traffic with similar pattern but destining for the same host can be excluded, such as http.

Definition 3.2 The *suspected traffic segment (STS)* is the effective traffic segment that contains at least one known bridge IP as destination address.

Based on the Tor specification, we have two observations. (1) Due to the feature of port locality, those connections within an effective traffic segment belong to one and the same application, which holds for widely used operating systems. (2) Any suspected traffic segment containing at least one bridge implies that all these connections may belong to the Tor bootstrapping session. Based on the two observations, we can conclude that if any destination IP contained among such packets belongs to a known bridge, all others can then be inferred to be of bridges too.

Figure 4 presents the principle of our detecting methods. Given an appropriate initial bridge set, the hidden bridges used in a controlled network can be detected and further blocked by recording and analyzing a series of suspected traffic segments. Furthermore, combined with our online bridge verification, we can avoid detection errors by real-time pruning, which will be discussed later.

3.2 Detecting algorithms

Based on our basic idea, we herein propose two detecting algorithms according to different available computing and storage resources, which also have different detecting effect.

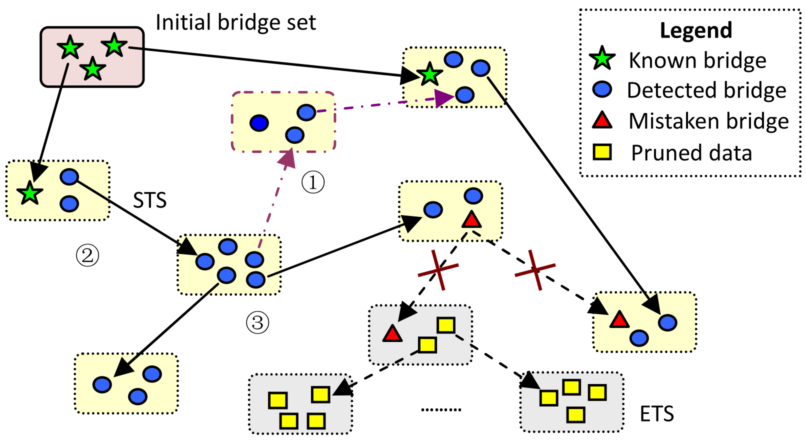

Fig. 4 Principle of the detecting methods

Table 1 Online detecting algorithm

```
Online detecting algorithm
1: Let $B$ denote initial bridge set
2: Given $\delta$ and $T$
3: for each extracted effective traffic segment $X$
4:     if $\exists y$ ($y \in X$ and $y \in B$)
5:         $X$ is a suspected traffic segment
6:         assign $\forall z$ ($z \in X$ and $z \notin B$ and verified) to $B$
7:     endif
8: end for
```

(1) Online detection algorithm

Given an initial bridge set, we first extract the current ETS from online traffic, which is denoted as a series of tuples of <*SrcIP*, *SrcPort*, *DstIP*, *DstPort*>. If any *DstIP* belongs to our current known bridge set (i.e. the current ETS is an STS), all other *DstIP*s can be inferred to be of suspected bridges too, which will be further confirmed employing online Tor protocol verification. The verified will then be added to the known bridge set, thereby extending the set. Otherwise, the current ETS will be discarded. In large networks, more computing resource is required for online detection to avoid packet loss. The online detecting algorithm is presented in Table 1.

(2) Offline detection algorithm

In order to increase the accuracy of bridge discovery, we can utilize the offline algorithm to find out all possible bridges by recording all extracted ETSs. Compared with online detecting algorithm, the uncorrelated ETSs must be recorded instead of being discarded. The reason resides in the fact that it may be correlated with the subsequent set of leaked bridges later. Therefore, the accuracy of the offline algorithm is higher than that of the online algorithm. The offline detecting algorithm is presented in Table 2.

Table 2 Offline detecting algorithm

Offline detecting algorithm

```
1: Let $B$ denote initial bridge set
2: Given $\delta$ and $T$
3: for each extracted effective traffic segment $X$
4:         record $X$
5: end for
6: do
7:     for all recorded effective traffic segment $X$
8:         if $\exists f$ ($f \in X$ and $f \in B$)
9:             $X$ is a suspected traffic segment
10:            assign $\forall g$ ($g \in X$ and $g \notin B$ and verified) to $B$
11:        end if
12:    end for
13: while $B$ has been extended
```

(3) Comparison between two algorithms

Compared with online detecting algorithm, offline detecting algorithm has higher detection rates and lower false-positive rates, given the same detecting parameters, which can be demonstrated in Fig. 4. For example, assume ETS ①–③ be extracted according to their sequence number. ETS ① cannot be correlated with initial bridge set and will be discarded for online detecting algorithm; whereas for offline detecting algorithm, ETS ① can be indirectly correlated with the initial bridge set by ETS ③ and ETS ②. On the other hand, for offline detecting algorithm, all ETSs must be recorded for future possible correlation, while online detecting algorithm only needs to keep tracking those ETSs within a specified sliding time window T. Therefore, offline detecting algorithm requires more storage resources, but relatively less computing capability. The tradeoff should be considered between the detection effectiveness and resource requirement.

3.3 Discussions

Our detecting methods can be tuned to achieve appropriate detection accuracy and false detection rates by adjusting some detection parameters, constructing appropriate initial bridge set and employing online bridge verification. Furthermore, some traffic volume issues are discussed.

(1) Detection parameters adjustment

The detection accuracy depends on multiple different parameters including the size of the sliding time window T and port interval δ. T should be determined by the interval that packets are sent locally and the jitter from the source to the detecting point. As the interval sent locally is trivial compared with the jitter, we can set T as the half of the maximum RTT to the detecting point in the controlled network. Usually, the packets sent by the same application in a short time interval will be assigned continuous source ports. However, the continuity may be interrupted by other interleaved network applications. Furthermore, some antivirus software such as Kaspersky may dynamically occupy the adjacent source ports for filtering users' traffic. δ will be set varying from 1 to 3 in our detecting algorithms. It is obvious that the smaller the δ, the smaller the detection rates will be. Therefore, we prefer larger value to achieve larger detection rates by combining with online bridge verification.

(2) Initial bridge set construction

In current Tor design, published bridge tuples are chosen from isolated bridge pools according to subscription means, where little correlation exists among different bridge pools. It would be better to construct the initial bridge set from every pool for full coverage. In addition, in practice the correlation between different pools may also be achieved in our methods with the aid of Tor users who may use multiple bridges from different sources. Furthermore, in order to work more efficiently, a large initial bridge set is recommended.

(3) Online bridge verification

Since the Tor client bootstrapping procedure may be interleaved with other processes accessing the network, there may exist some ETSs containing a few faked bridge IPs, thereby causing detection errors and subsequent error spread. We herein propose an online bridge verification approach. The suspected bridge can be confirmed if a one-hop Tor circuit targeted at it can be successfully constructed. Otherwise, the suspected bridge will be discarded. The efficiency of such online verification approach depends on how often the verification will be conducted, which is determined by the total number of available bridges and false-positive rates. Since the number of bridges is quite limited and false-positive rates of our algorithms are low as demonstrated in Sect. 5, such an online verification approach will not degrade the performance of detection methods much, while reducing false-positive errors.

(4) Traffic volume issues

To address the large volume of traffic at gateways, we on one hand, only monitor outbound TCP SYN packets from controlled networks. On the other hand, we exclude the most common traffic with the similar pattern as ETSs but destining for the same host, such as http, thus greatly reducing the computing and storage requirement. Therefore, our detecting methods can still achieve high performance even for a large managed network.

In contrast to the online method, the offline method requires more storage for collecting all of the ETSs and more time to find bridges as well. However, the offline detection can be implemented in a non-timely way. Meanwhile, its performance can be further improved by utilizing parallel algorithms.

4 Theoretical analysis

In order to analyze the effectiveness and efficiency of our detecting methods, we first build the probabilistic detecting model and then give the corresponding statistical formula. For brevity, we focus on the publishing strategy based on one bridge pool, and adopt the assumption that the three bridges contained in one tuple are chosen from the same Tor bridge pool with equal probability. For consistency, we also called the chosen bridge tuple as an ETS. We denote the size of initial bridge set as n and the size of Tor bridge pool as m. The online and offline detecting models are presented, respectively.

4.1 Probabilistic model of online detection

Online detection means that if the current ETS has intersection with the known bridge set B (i.e. the ETS is an STS), B can be extended as $B \cup ETS$. We need to calculate the expected mean size of B after having online processed x ETSs, which can be used to evaluate the impact of our online detecting method.

Let f_k^x denote the probability that the size of B will be k after having processed the xth ETS. In order to get a set with k bridges after having processed the xth ETS, there are exactly four cases concerned.

(1) Case I means that all the three bridges included in the xth ETS are already in B, thereby keeping the size of B unchanged as k. The possibility of Case I donated as p_1 can be calculated correspondingly as follows:

$$p_1 = \frac{\binom{k}{3}}{\binom{m}{3}} \tag{1}$$

(2) Case II means that all three bridges included in the xth ETS do not belong to B. Therefore, according to our online detecting method, the size of B still remains unchanged. The possibility of Case II donated as p_2 can be calculated correspondingly as follows:

$$p_2 = \frac{\binom{m-k}{3}}{\binom{m}{3}} \tag{2}$$

(3) In Case III, two of three bridges contained in the ETS belong to B. It is obvious that the size of B (i.e. $k-1$) can be increased by one. The possibility of Case III donated as p_3 can be calculated correspondingly as follows:

$$p_3 = \frac{\binom{k-1}{2}\binom{m-k+1}{1}}{\binom{m}{3}} \tag{3}$$

(4) In Case IV, only one of three bridges contained in the ETS belong to B. The size of B (i.e. $k-2$) can be increased by two. The possibility of Case IV donated as p_4 can be calculated correspondingly as follows:

$$p_4 = \frac{\binom{k-2}{1}\binom{m-k+2}{2}}{\binom{m}{3}} \tag{4}$$

Based on the above analysis, we can use the recurrence formula (5) to calculate the possibility f_k^x that the size of known bridge set B is k after having processed the xth ETS.

$$f_k^x = f_k^{x-1} p_1 + f_k^{x-1} p_2 + f_{k-1}^{x-1} p_3 + f_{k-2}^{x-1} p_4 \tag{5}$$

The initial conditions can be described as follows:

$$\begin{cases} f_l^x = 0 \quad (l < n) \\ f_l^0 = 0 \quad (l \neq n) \\ f_n^0 = 1 \end{cases} \tag{6}$$

In order to calculate the expected mean size of B after having processed the xth ETS, the possibility f_k^x must be normalized to get the probability F_k^x by utilizing the following approach:

$$F_k^x = \frac{f_k^x}{\sum_{l=n}^{m} f_l^x} \tag{7}$$

Let N^x denote the expected mean size of B after having processed the xth ETS. We can calculate N^x as follows:

$$N^x = \sum_{l=n}^{n+2x} l \cdot F_l^x \tag{8}$$

The analysis result based on formula (8) will be presented in Sect. 5.

4.2 Probabilistic model of offline detection

Offline detection means that if any recorded ETS has intersection with the known bridge set B (i.e. the ETS is an STS), B can be extended as $B \cup ETS$. Such processing must be iterated many rounds for all recorded ETSs until B becomes unchanged. Such a problem can be modeled as sampling with replacement, given a subset B with n balls and its superset A with m different balls. We aim to calculate the expected mean size of B after having recorded x samples.

Let N^x denote the expected mean size of B after having recorded x samples. Among the all x samples, there are four scenarios. Case I means that there are a samples where all three sampled balls belong to the initial set B. Case II means that there are b samples where two sampled balls belong to the initial set B while the rest belongs to the set $A - B$. Case III means that there are c samples where one sampled ball belongs to the initial set B while the other two balls belong to the set $A - B$. Case IV means that there are d samples where all three sampled balls belong to the set $A - B$. In a sampling procedure with x samples, we have $a + b + c + d = x$.

Let $M_{a,b,c,d}$ denote the expected mean size of set B for a certain number x samples, among which the number of corresponding samples for different scenarios is a, b, c, and d, respectively. The expected mean size of set B can be calculated as follows:

$$\begin{aligned} N^x = \sum_{a,b,c,d} \frac{x!}{a!b!c!d!} \left(\frac{\binom{n}{3}}{\binom{m}{3}}\right)^a \cdot \left(\frac{\binom{n}{2}\binom{m-n}{1}}{\binom{m}{3}}\right)^b \cdot \left(\frac{\binom{n}{1}\binom{m-n}{2}}{\binom{m}{3}}\right)^c \\ \cdot \left(\frac{\binom{m-n}{3}}{\binom{m}{3}}\right)^d \cdot [M_{a,b,c,d}] \end{aligned} \tag{9}$$

5 Evaluation

We evaluate the effectiveness and efficiency of the proposed methods through simulations and real-world experiments. We first validate the models and verify the effectiveness of our methods by simulation. We then analyze the correlation between the practical bridge tuples published by Tor via email and https. Finally, we present the real-world experiment and its result.

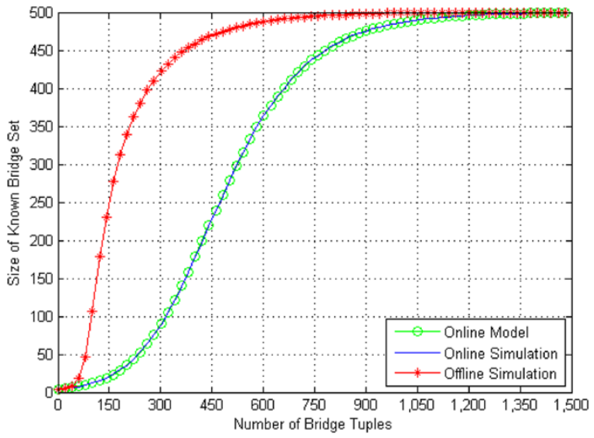

Fig. 5 Expected mean size of known bridge set

5.1 Verifying the effectiveness by simulation

We present the Monte Carlo simulation of both online and offline methods. In each simulation, the total number of the bridges available is m and the size of the initial bridge set is n. Each time we choose three bridges in the pool with equal probability to construct an ETS. The simulation result is shown in Fig. 5, where m is set to 500 and n is set to 3.

From Fig. 5, we can find that the simulation result of online detecting method discussed well fits the theoretical result of online model represented by formula (8), which validates the built model. Since the computing overload is very high, we conducted simulation of offline detection instead of theoretical computing to evaluate the effectiveness of offline method.

Based on the simulation results of both methods, most bridges can be detected after having processed enough ETSs, verifying the effectiveness of our methods. In addition, given the same number of ETSs, the size of known bridge set of offline detecting method is significantly larger than that of online detecting method. As shown in Fig. 5, when the number of ETSs reaches 300, the number of known bridges for offline detection is about 410 (i.e. 82 % of total bridges in the pool), while the number of known bridges for online detection is less than 80 (16 %); when the number of ETSs reaches 800, the number for offline detection is about 495 (99 %), while the number for online detection is about 450 (90 %). The results also imply that for real-world detection, our methods can work quite efficiently either when the initial bridge set is large enough or sufficient Tor users have been monitored.

5.2 Analyzing the correlation between published tuples

In order to verify the high correlation between published bridge tuples, we conducted Monte Carlo simulation based on practical traces of published bridges by Tor via email and https, given different size of initial bridge sets. We first applied for 1000 Gmail/Yahoo accounts, and then sent request emails using each account to bridges@torproject.org. After that, we received an automated response containing a

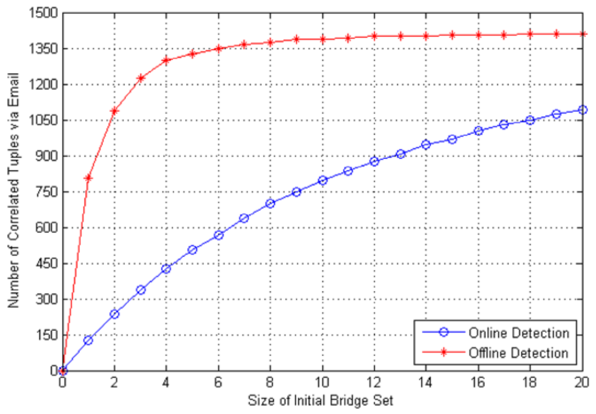

Fig. 6 Correlation between published tuples via email

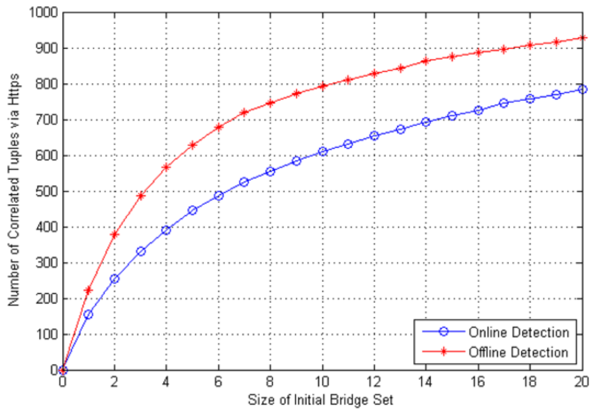

Fig. 7 Correlation between published tuples via https

bridge tuple of three bridges, where one response was limited per email address one day. In addition, we also obtained bridge tuples by visiting https://bridges.torproject.org via 500 different PlanetLab nodes, where the answered bridges from that page was distributed based on the IP address of the requesting user. In this way, we have obtained 1391 bridge tuples through different email accounts and 1068 tuples through different https source IPs, respectively.

In each round of simulation, we first sampled an initial bridge set with a specified size, and then counted how many tuples could be correlated using our detecting methods. The average number of correlated tuples will be output as the simulation result. Figures 6 and 7 present the simulation results based on email and https dataset, respectively.

The results both demonstrate that with the increasing size of initial bridge set, the number of correlated tuples increases rapidly, verifying the high correlation between published bridge tuples. We can also find that offline method can correlate more tuples than the online method, given the same size of initial bridge set.

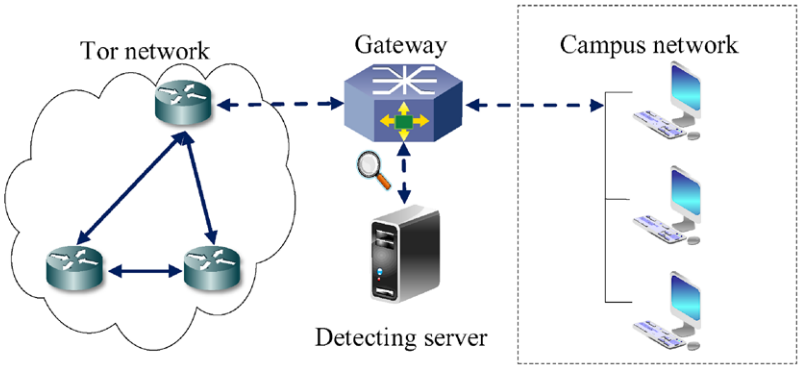

Fig. 8 Experimental setup

5.3 Real-world experiment

We conducted a real-world experiment to detect bridges using our detecting methods.

(1) Experimental setup

The experimental setup was shown in Fig. 8, where we deployed our detecting point at the gateway between our campus network and Internet. All traffic transferred through the gateway will be mirrored to our detecting server. We distributed bridge tuples obtained via email to Tor users in the campus network for evaluation, which amount to 248 different bridges. The detecting server was responsible for extracting the ETSs, according to the features of Tor's outbound SYN packets with consecutive source ports, and conducting further detections.

(2) Experimental procedure

We adopted an initial bridge set with the size of 6 by randomly choosing two tuples obtained. According to our detecting methods proposed in Sect. 3, the online and offline detecting methods were first used, respectively, to extract the traffic segments within a short time interval T with the same source IP address, consecutive source ports and different destination IP addresses. We set T to 50 ms and δ to 1–3 in our experiments. We then analyze the ETSs for online and offline detecting methods, respectively.

- *Online detection*. If the current ETS has intersection with the known bridge set, all suspected bridges contained in the ETS are detected and further verified employing the online verification. Only the verified bridges will be added to the known bridge set.
- *Offline detection*. We first recorded all ETSs extracted so far. Our offline detecting algorithm is triggered whenever 10,000 ETSs are newly recorded. The procedures of the detection and verification were similar to that for online detection, except that such processings were iterated many rounds for all recorded ETSs until the known bridge set became unchanged.

(3) Experimental result

We present the detection rates and false-positive rates for online and offline detection methods, respectively. The detection rates were measured as the number of

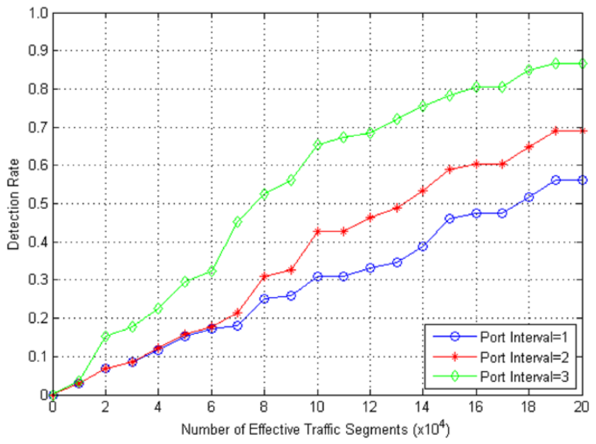

Fig. 9 Online detection rates with different port intervals

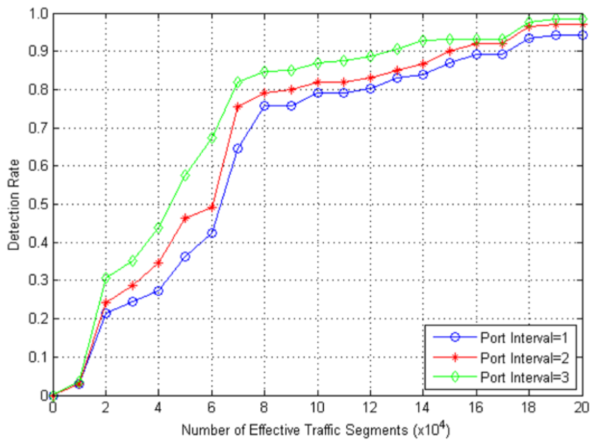

Fig. 10 Offline detection rates with different port intervals

detected bridges over the number of bridges (248) we published. Figures 9 and 10 present the detection rates of online and offline detecting method respectively, where the port interval parameter δ is set varying from 1 to 3. The empirical results imply that the larger the number of effective traffic segments, the higher the detection rates may achieve. Furthermore, for a fixed number of traffic segments, the larger port interval will lead to higher detection rates. When the port interval is set to 3, we can achieve a detection rate of 86.7 % for online method and a 98.4 % detection rate for offline detection after having extracted 190,000 ETSs.

Compared with the online method, the detection rates of the offline method are significantly higher, which is also consistent with the analysis result in Sect. 4. Furthermore, the port interval has less impact on detection rates for the offline method than for the online method.

Figures 11 and 12 present the false-positive errors of online and offline methods, respectively.

Given a fixed number of traffic segments, the larger the port interval parameter is, the more the false-positive errors are. Compared with online detecting method,

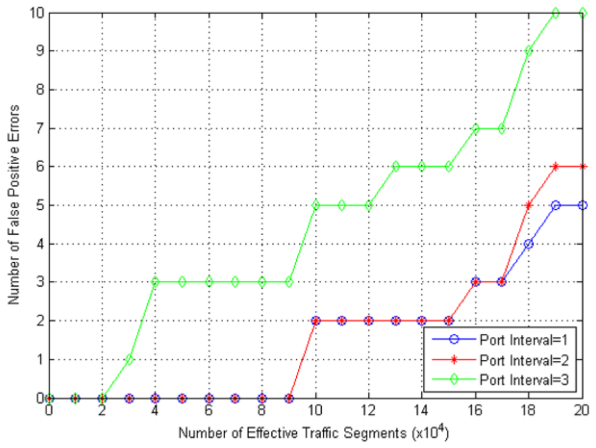

Fig. 11 False-positive errors of online method

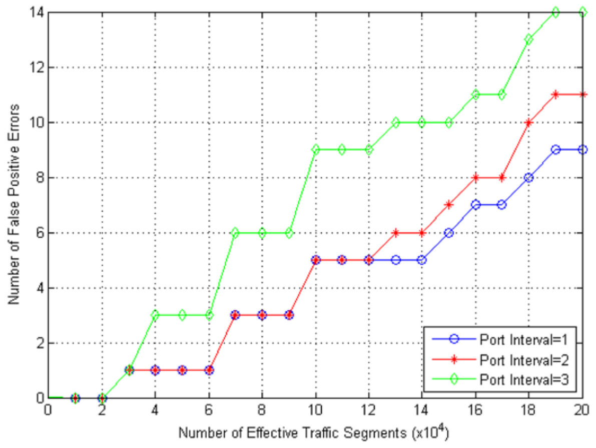

Fig. 12 False-positive errors of offline method

the false-positive errors of offline method are significantly more than that of online method. Although the offline method can achieve higher detection rates by processing recorded ETSs repeatedly, those false-positive ETSs will be exploited for extending the known bridge set, resulting in higher false-positive errors.

The false-positive rates are measured as the number of false-positive errors over the number of STSs. Figure 13 presents false-positive rates of online and offline methods with varying port intervals. From the figure, we can find that the false-positive rates of online method are higher than that of offline method, whereas the number of false-positive errors of offline method is higher. Furthermore, for the online method, the lowest false-positive rate can be achieved with the port interval set to 2. The reason resides in the fact that although a larger port interval may cause more false-positive errors, it leads to much more STSs. As a result, the lower false-positive rate is achieved.

In addition, we find that quite a few users in practice use more bridges than the default value (i.e. three), which results in higher detection rates.

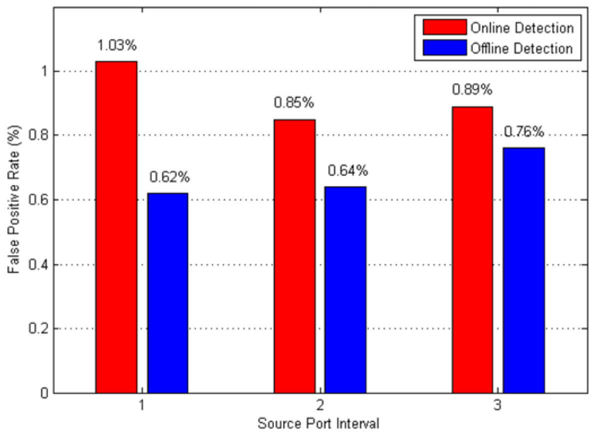

Fig. 13 False-positive rates of online and offline detection

6 Countermeasures

In this section, we propose some countermeasures to our attacks.

From the perspective of Tor users, the best alternative is to use only one bridge each time when connecting to Tor network, where only one SYN packet will be sent. As a result, there is no correlation between the chosen bridge and those bridges in the leaked bridge set. Avoiding using leaked bridges any longer is another countermeasure. Since new bridges contained in ETSs are detected by correlating with leaked bridges, any traffic segment without leaked bridges inside will not be recognized as the STS, eliminating the possibility of being detected.

From the perspective of Tor network, there are also some alternatives which can be used. Intuitively, the Tor software itself can be modified to use randomly selected source ports, for the purpose of avoiding port continuity. However, such a traffic pattern is inherent due to the Tor client's bootstrapping and can also be easily recognized by monitoring connection characteristics such as initiated sessions and port pairs, which have been widely used for P2P traffic detection [5]. In addition, in order to avoid distributing blocked bridges, the bridge publishing mechanism should be further improved. As a result, those non-leaked bridges will not be detected by correlation. Furthermore, distributing bridges mixed with fake ones is an effective idea to degrade the correlation, as the wrong addresses may mislead the attacker and render the correlation inefficient.

In order to further survive blocking, the architecture of Tor should be changed to pure P2P model. In this way, the client acts as a bridge, and not only requests anonymous services from other Tor bridges, but also provides services to others. Therefore, it is impossible for an attacker to block all distributed bridges in a P2P fashion due to the vast number of clients or bridges.

7 Related work

Recently, a number of attacks have been presented against anonymous communication systems, including anonymity degrading, communication relationship confirmation and hidden service location etc. In addition, some blocking-resistance mechanisms have also been presented.

Murdoch et al. [6] presented a low-cost traffic analysis technique to degrade the anonymity of Tor, which allowed an outside observer to infer which routers were being used to relay a circuit's traffic. Given some malicious routers, Bauer et al. [7] presented a low-resource attack by attracting Tor traffic. Some attacks exploit Tor's design features to compromise the anonymity of Tor clients. Evans et al. [8] proposed a practical congestion attack on Tor using long paths, and presented the countermeasures by modifying Tor protocol to avoid constructing long circuits. Abbott et al. [9] presented the browser-based attack against Tor exploiting only JavaScript. Manils et al. [10] presented three attacks targeting BitTorrent users on top of Tor to reveal their real IP addresses.

Some attacks have been presented to confirm the communication relationship among users. Zhu et al. [11] proposed the scheme using mutual information for the traffic similarity measurement, while Levine et al. [12] utilized a cross-correlation technique for the traffic similarity measurement. Wang et al. [13] proposed an active watermarking scheme by manipulation of interpacket delays to confirm the communication relationship. They analyzed the tradeoffs between the true positive rate, the maximum timing perturbation added by adversaries, and the number of packets needed to successfully decode the watermark. To make the watermark more robust against flow transformation such as packet losses, insertions and repacketization, Wang et al. [14] proposed interval centroid-based watermarking scheme. Pyun et al. [15] proposed interval-based watermarking scheme for the purpose of tracing the traffic in the presence of repacketization. To achieve more invisibility, Yu et al. [16] proposed a DSSS-based traceback technique aiming to achieve traceback invisibility. Houmansadr et al. [17] proposed a new non-blind watermarking scheme called RAINBOW that is able to use delays hundreds of times smaller than existing watermarks. The replay attack [18] exploits the fundamental design of Tor by duplicating cells to disrupt the AES counter at middle and exit onion routers, as Tor uses the counter mode of Advanced Encryption Standard (AES-CTR) for encryption and decryption of cells at onion routers. Ling et al. [19] presented a new cell counter-based attack against Tor to confirm anonymous communication relationship among users efficiently.

A few attacks have been proposed to locate the hidden service offered by Tor. Overlier et al. [20] presented an attack that used one compromised router to generate false resource claims to attract traffic. The approach is based on the traffic timing similarity to correlate circuits, thereby locating the hidden servers in the Tor network. Murdoch et al. [21] investigated an attack to reveal hidden servers in Tor by exploiting the fact that the clock deviations of a target server should be consistent with the server's load. Zander et al. [22] proposed an improved skew estimating method to achieve and maintain the synchronization, despite network jitter.

In order to avoid being blocked, some blocking-resistance mechanisms for anonymous communication have been presented. Danezis et al. [23] identified fundamental flaws in two dominant classes of anonymity revocation systems for bypassing anonymity revocation. McLachlan et al. [24] identified three key architectural shortcomings of the bridge design, showing the risks of serving while surfing. Köpsell et al. [25] proposed a blocking-resistant, practical and usable system for anonymous web surfing. Recently, Ling et al. [26] proposed two categories of bridge discovery approaches: (i) enumerating bridges from bridge https and email servers by employing large numbers of PlanetLab nodes, and (ii) inferring bridges by deploying malicious Tor middle routers.

Existing blocking mechanisms for anonymous communication focus on anonymity revocation, while our proposed blocking methods aim to deny clients from accessing Tor network, thus eliminating any potential anonymous communications. Furthermore, compared with the work in [26], our methods focused on the detection of bridges used only in a controlled network instead of enumerating all the bridges in the Tor network, so our methods can work effectively with fewer resources.

8 Conclusions

In this paper, we proposed blocking methods against Tor exploiting the vulnerabilities of the Tor bridge design. According to the computing and storage resources available, we present both the online and offline detecting methods. In order to evaluate the effectiveness of our methods, we first built the detecting models and gave the probability statistic formula for calculating the expected mean number of detected bridges. Under the assumption of equal probability sampling from a bridge pool containing 500 different bridges, we secondly conducted thoroughly Monte Carlo simulations on both online and offline detecting methods. The simulation results well fit our theoretical models, and demonstrate that both detecting methods can work effectively. Thirdly, we obtained 1391 bridge tuples through different Gmail/Yahoo accounts and 1068 tuples through different https source IPs, respectively. The result of simulation on such dataset demonstrates that there exists high correlation among bridge tuples published by Tor network, which proves the feasibility of our methods. Finally, we conducted real-world experiments by deploying our detecting server at the gateway between our campus network and Internet. By configuring optimized detecting parameters, we can achieve a detection rate of 86.7 % with a 0.85 % false-positive rate for online detection, and a 98.4 % detection rate with a 0.62 % false-positive rate for offline detection, validating the impact of our methods.

To make up the flaws in Tor's current block-resistance mechanism, we also provide some countermeasures from the perspective of Tor network and users respectively, such as avoiding using leaked bridges for Tor users, and binding randomly selected source ports for Tor software itself. Furthermore, pure P2P model is suggested as the future architecture for the Tor network.

Acknowledgements This work is supported by National Key Basic Research program of China under Grants No. 2010CB328104, National Natural Science Foundation of China under Grants No. 60903162, No. 60903161, No. 61070161 and No. 61003257, China National Key Technology R&D Program under Grants No. 2010BAI88B03 and No. 2011BAK21B02, China Specialized Research Fund for the Doctoral Program of Higher Education under Grants No. 20110092130002, Jiangsu Provincial Natural Science Foundation of China under Grants No. BK2008030, Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No. BM2003201 and Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grants No. 93K-9.

References

1. Dingledine R, Mathewson N, Syverson P (2004) Tor: the second-generation onion router. In: Proceedings of the 13th USENIX security symposium, San Diego, CA, USA, pp 303–320
2. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–90. doi:10.1145/358549.358563
3. Dingledine R, Mathewson N. Design of a blocking-resistant anonymity system DRAFT. https://svn.torproject.org/svn/projects/design-paper/blocking.html
4. Larsen M, Gont F (2010) Transport protocol port randomization recommendations. Internet-draft, 31 May 2010
5. Karagiannis T, Broido A, Brownlee N, Claffy C, Faloutsos M (2004) Is P2P dying or just hiding? In: Proceedings of IEEE global telecommunications conference (GLOBECOM), Dallas, TX, USA, pp 1532–1538
6. Murdoch SJ, Danezis G (2005) Low-cost traffic analysis of Tor. In: Proceedings of IEEE symposium on security and privacy (S&P), Oakland, CA, USA, pp 183–195
7. Bauer K, McCoy D, Grunwald D, Kohno T, Sicker D (2007) Low–resource routing attacks against tor. In: Proceedings of the 2007 ACM workshop on privacy in the electronic society (WPES)
8. Evans NS, Dingledine R, Grothoff C (2009) A practical congestion attack on Tor using long paths. In: Proceedings of the 18th USENIX security symposium (security), Montreal, Canada, August 10–14
9. Abbott T, Lai K, Lieberman M, Price E (2007) Browser-based attacks on Tor. In: Proceedings of the 7th international symposium on privacy enhancing technologies (PET), Ottawa, ON, Canada, pp 184–199
10. Manils P, Abdelberri C, Blond S, Mohamed AK, Castelluccia C, Legout A, Dabbous W (2010) Compromising Tor anonymity exploiting P2P information leakage. arXiv:1004.1461
11. Zhu Y, Fu X, Graham B, Bettati R, Zhao W (2004) On flow correlation attacks and countermeasures in mix networks. In: Proceedings of the workshop on privacy enhancing technologies (PET), Toronto, ON, Canada, pp 207–225
12. Levine BN, Reiter MK, Wang C, Wright M (2004) Timing attacks in low-latency mix systems. In: Proceedings of financial cryptography (FC), Key West, FL, USA, pp 251–265
13. Wang X, Reeves DS (2003) Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In: ACM conference on computer and communications security, Washington, DC, USA, pp 20–29
14. Wang X, Chen S, Jajodia S (2007) Network flow watermarking attack on low-latency anonymous communication systems. In: Proceedings of IEEE security and privacy symposium (S&P), Oakland, CA, USA, May 2007, pp 116–130
15. Pyun Y, Park Y, Wang X, Reeves DS, Ning P (2007) Tracing traffic through intermediate hosts that repacketize flows. In: Proceedings of IEEE conference on computer communications (INFOCOM), Anchorage, AK, USA, May 2007, pp 634–642
16. Yu W, Fu X, Graham S, Xuan D, Zhao W (2007) DSSS-based flow marking technique for invisible traceback. In: Proceedings of IEEE security and privacy symposium (S&P), Oakland, CA, USA, May 2007, pp 18–32
17. Houmansadr A, Kiyavash N, Borisov N (2009) RAINBOW: a robust and invisible non-blind watermark for network flows. In: Proceedings of the 16th annual network & distributed system security symposium (NDSS)
18. Pries R, Yu W, Fu X, Zhao W (2008) A new replay attack against anonymous communication networks. In: Proceedings of IEEE international conference on communications (ICC), Beijing, China, pp 1578–1582
19. Ling Z, Luo J, Yu W, Fu X, Xuan D, Jia W (2009) A new cell counter based attack against Tor. In: Proceedings of the 16th ACM conference on computer and communications security (CCS)
20. Overlier L, Syverson P (2006) Locating hidden servers. In: Proceedings of IEEE symposium on security and privacy (S&P), Berkeley, CA, USA, pp 100–114
21. Murdoch SJ (2006) Hot or not: revealing hidden services by their clock skew. In: Proceedings of the 13th ACM conference on computer and communications security (CCS), Alexandria, VA, USA, pp 27–36
22. Zander S, Murdoch S (2008) An improved clock-skew measurement technique for revealing hidden services. In: Proceedings of the 17th USENIX security symposium (security)
23. Danezis G, Sassaman L (2008) How to bypass two anonymity revocation schemes. In: Privacy enhancing technologies (PETS 2008). Lecture notes in computer science, pp 187–201. doi:10.1007/978-3-540-70630-4_12

24. McLachlan J, Hopper N (2009) On the risks of serving whenever you surf: vulnerabilities in Tor's blocking resistance design. In: Proceedings of the ACM conference on computer and communications security, Chicago, IL, USA, 9–13 November 2009, pp 31–40
25. Köpsell S, Hillig U (2004) How to achieve blocking resistance for existing systems enabling anonymous web surfing. In: Proceedings of the 2004 ACM workshop on privacy in the electronic society (WPES), pp 47–58
26. Ling Z, Luo J, Yu W, Yang M, Fu X (2012) Extensive analysis and large-scale empirical evaluation of Tor bridge discovery. In: Proceedings of the 31th IEEE international conference on computer communications (INFOCOM), Orlando, FL, USA, 25–30 March 2012, pp 2381–2389