PAPER Special Section on Information and Communication System Security

A Novel Protocol-Feature Attack against Tor's Hidden Service

Rui WANG[†], ***Student Member***, **Qiaoyan WEN**[†], ***Nonmember***, **Hua ZHANG**[†a)], ***Member***, ***and*** **Xuelei LI**[†], ***Nonmember***

SUMMARY Tor is the most popular and well-researched low-latency anonymous communication network provides sender privacy to Internet users. It also provides recipient privacy by making TCP services available through "hidden service", which allowing users not only to access information anonymously but also to publish information anonymously. However, based on our analysis of the hidden service protocol, we found a special combination of cells, which is the basic transmission unit over Tor, transmitted during the circuit creation procedure that could be used to degrade the anonymity. In this paper, we investigate a novel protocol-feature based attack against Tor's hidden service. The main idea resides in fact that an attacker could monitor traffic and manipulate cells at the client side entry router, and an adversary at the hidden server side could cooperate to reveal the communication relationship. Compared with other existing attacks, our attack reveals the client of a hidden service and does not rely on traffic analysis or watermarking techniques. We manipulate Tor cells at the entry router to generate the protocol-feature. Once our controlled entry onion routers detect such a feature, we can confirm the IP address of the client. We implemented this attack against hidden service and conducted extensive theoretical analysis and experiments over Tor network. The experiment results validate that our attack can achieve high rate of detection rate with low false positive rate.

key words: *hidden service, Tor, protocol-feature, anonymity communications*

1. Introduction

Tor [1], as the realization of second generation onion routing arguably is the most popular and well-researched low-latency anonymity network, providing sender privacy to its users. Nonetheless, recipient privacy is equally important and required. Tor's hidden service [3] provides recipient privacy by making TCP services available, allowing users not only to access information anonymously but also to publish information anonymously. Tor was first deployed in 2003[2], and the hidden services were released in early 2004. By May 2015, there are more than 6700 onion routers in Tor network.

Both sender privacy and recipient privacy are provided in hidden service, it allows people to deploy a TCP Server that can provide hidden services (e.g., a web server, SSH server, etc.) over Tor without exposing their real IP address to users. Nonetheless, there exist vulnerabilities that can be utilized by adversaries to degrade the anonymity. Existing works [4]–[6], [8], [12] have been conducted to investigate attacks degrading the communication over Tor's hidden service. These attacks can be categorized into two classes: passive attacks and active attacks. Passive attacks [5], [6], [8], [12] will record the traffic passively and do traffic analysis. This type of technique needs a period of traffic observation. The first passive attack [5] was proposed by Overliver and Syverson, they assumed that the adversary controlled the entry router at the hidden server side and correlated the traffic pattern between malicious client and the entry router. The passive attacks based on traffic analysis may suffer a high rate of false positives due to various factors, and all of these attacks focus on locating the hidden server's real IP address. Active attacks mainly refer to the active watermarking technique based attacks, this technique can reduce attack lasting time and improve attack success rate. Though watermarking techniques have been widely applied into attacks against Tor, there are rare studies on active attacks against hidden service. Murdoch presented the first active attack [4]. A malicious client periodically sent bulk data to the hidden server that lead its CPU temperature to increase, and the adversary at the server side can locate the hidden service base on detecting the temperature change. However, the hidden circuit between the client and the hidden server is bi-direction anonymous, and hidden services are not secure enough to protect the sender privacy.

In this paper, we present a novel protocol-feature attack against hidden service. The attack requires three phases. In the first phase, our controlled entry router keeps monitoring the traffic and reporting related cells' information to a controlled server, denoted as central server. We aim to find a combination of cells corresponding to the client side protocol-feature. In the second phase, our entry router manipulates a special cell and relays it along the circuit to the hidden server, which will cause decryption error at the hidden server. Then, the hidden server tears down the circuit by sending a destroy cell. During this process, we record three specific timestamps. In the third phase, the central server filters out useless information based on the client side protocol-feature, searching the remaining hidden information again and trying to find a circuit which can detect the destroy cell and contain three timestamps. Then, the central server checks whether the three timestamps are appropriate. If such a circuit was found, we can confirm that our controlled router is chosen by the client as the entry router, and

Manuscript received March 29, 2015.
Manuscript revised October 10, 2015.
Manuscript publicized January 13, 2016.

[†]The authors are with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, 100876 China.
a) E-mail: zhanghua_288@bupt.edu.cn

DOI: 10.1587/transinf.2015ICP0001

we can locate the client.

The main contribution of this paper include: first, we present an attack against hidden service that can reveal the client of hidden service, and our method achieves high detection rate with low false positive rate. Second, we find the protocol-feature at the client side, and we implemented this attack and performed extensive theoretical analysis and real-world experiments over Tor. Third, we only make a small change to the source code and configuration file of Tor to build the system, and it is easy to deploy. Fourth, we only need to control the entry router at the client side and only use a few cells to locate the client.

The rest of this paper is organized as follows: in Sect. 2, we review the previous related works. In Sect. 3, we introduce basics of Tor and hidden service, and describe the detailed circuit creation procedure of hidden service protocol. In Sect. 4, we elaborate our attack. In Sect. 5, we theoretically analyze the feasibility of our attack. In Sect. 6, we show our experiment results and data analysis. In Sect. 7, we discuss complicated cases of hidden service. In Sect. 8, we give our conclusions.

2. Related Work

A good review of anonymous communication and mix systems can be found in [2], [21]. Extensive studies have been conducted to investigate attacks degrading the communication over Tor's hidden service. Existing attacks against anonymous communication can be categorized into two groups: active attacks and passive attacks. Murdoch first presented the active attack in [4], the malicious client periodically sent bulk data to the hidden server that lead its CPU temperature to increase, and the adversary at the server side can locate the hidden service base on detecting the temperature change. Overlier et al. presented the first traffic analysis based passive attack in [5]. The malicious client and entry router uploaded related information. Then, the system did timing analysis on the information to find the matching pattern, and locate the hidden server. However, this traffic analysis based system suffers a high rate of false positives due to Internet traffic dynamics, performance of Tor nodes, the large number of cells in traffic, etc. Entry guard nodes [16] which used to prevent the traffic analysis attack against the current version of Tor were added to the hidden services.

Biryukov Alex et al. [6] expose flaws both in the design and implementation of Tor's hidden services that allow an attacker to measure the popularity of arbitrary hidden services, take down hidden service and deanonymize hidden services. Pries et al. [14] presented a new relay attack against Tor. They manipulated a cell at the entry router, an accomplice at the exit router can recognize this cell, and then confirm communication relationships quickly and accurately. Zhen Ling et al. [7] presented the protocol-level attacks against Tor. They analyzed the detail process of Tor communication, and found that the cell type and numbers of each type cell during the communication is fixed. They apply the attack in [14] and develop another three case to confirm communication relationships. Based on the protocol-level attack, they designed and implemented a hidden server discovery system in [8]. The system controlled some high bandwidth onion routers in Tor network, and connected to their controlled Rendezvous point to hidden server. They assumed that their controlled routers could be selected as the entry router at the server side and submitted related cell information to the central server. Then the system could use a timing analysis to identify the hidden server.

Theoretical analysis and fundamental research provides us a better understanding of Tor network. For example, Loesing et al. [22] investigated the statistical data measurement in Tor. They provide different methods to measure data. Mohajeri et al. [23] analyzed the new obfuscation protocol of Tor, and an application of this protocol as SkypeMorph. Zhang et al. [25] investigated the anonymity of different anonymous communication systems. They modeled the P2P anonymous communications and took quantitative analysis of anonymity by information theory with entropy. Studies [9]–[11], [19], [20] are also attacks against Tor. We learn theoretical basis and practical application of different types of attacks against Tor from these studies which help us deeply understand mechanism of Tor, and we attempt to make improvement and innovation based on these studies and apply into hidden service.

3. Background

3.1 Components

Tor is a distributed global network, adopting centralized directory management structure. Figure 1 illustrates the Tor network structure. Four types of entities [1] are involved in the Tor network: *client*, *server*, *onion routers* and *directory servers*:

- *Alice* (i.e. Client). The client runs a local software called onion proxy (OP) to establish data transmission circuit to Tor network.
- *Bob* (i.e. Server). The server runs TCP applications such as a web service and anonymously communicates with Alice over Tor.
- *Onion Routers* (OR). The onion routers construct

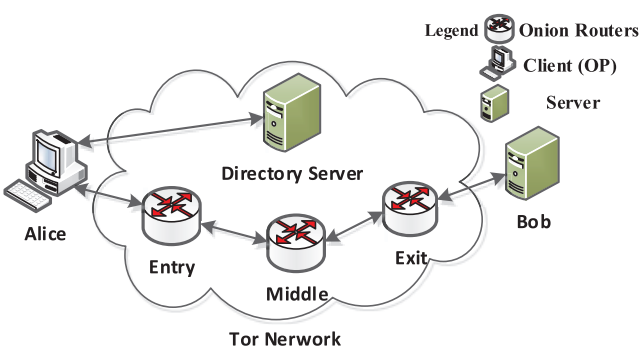

Fig. 1 Structure of Tor network.

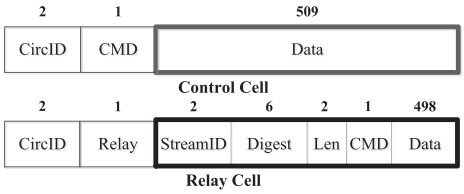

Fig. 2 Tor cell format.

anonymous communication links and relay cells between Alice and Bob. A circuit is a path of three onion routers by default through the Tor network, namely entry, middle and exit, respectively. Transport Layer Security (TLS) connections are used for the overlay link encryption between two onion routers. The application data is packed into 512 bytes equal size cells transmitted through TLS connections.

- *Directory servers*. Directory servers are in the uppermost layer network topology server and hold authoritative information such as public key of onion routers.

Among the three ORs, only the entry router communicates directly with the client, and only the exit router communicates with the Internet destination that the client is connecting with, but no OR can link a client to a destination; this is how a clients privacy is maintained in Tor.

Cell format

Figure 2 illustrates the cell format of Tor. All cells are in fixed-size (512 bytes) and have a three-byte header which is not encrypted in the onion-like fashion so that the intermediate Tor routers can see this header and forward the cell based on the header information, and other bytes are encrypted. The first two bytes is the circuit ID as the unique identification of a link, and the third byte is used to indicate the command type of this cell. The cell can be categorized into control cell and relay cell.

3.2 How Tor Works

A Tor client runs an OP locally. The client first downloads a list of routers from the directory server and randomly selects several ORs to build a circuit [17]. The client selects an appropriate exit onion router (OR3), which should have an exit policy supporting the relay of the TCP stream from the client. Then, the client selects a proper entry router, a middle router and creates a circuit incrementally one hop at a time. If multiple circuits use the same two ORs in sequence, they will share a single connection between the two ORs. Figure 3 illustrates the creation procedure of a standard three-hop circuit. Alice first establishes a TLS connection with OR1. Then, she sends a *CELL_CREATE* cell to OR1, using *Diffie-Hellman* protocol [1] to negotiate a base key $K_1 = g^{xy}$ with OR1. If Alice successfully obtained the entry router's public key, the first hop of this circuit is created. Similarly, the client extends the circuit to include the second hop and the third hop of the circuit.

Once the circuit is established, Alice sends a *RELAY_COMMAND_BEGIN* cell, which is encrypted in onion fashion as $\{\{\{Begin < IP, Port >\}_{k1}\}_{k2}\}_{k3}$. When the cell arrives at the exit router, the three layers of skin are removed at each OR. After the exit router removes the last skin by decryption, it will build a TCP connection to Bob and relay a *REALY_COMMAND_CONNECTED* cell back to Alice. The traffic flow between the exit router and Bob will be transmitted in clear type. Then, the circuit is built up, and Alice starts transmitting data with Bob.

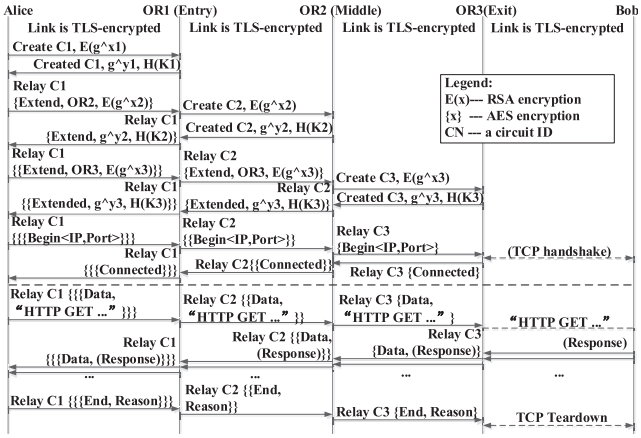

Fig. 3 Procedure of circuit creation on Tor.

3.3 How Hidden Service Works

Tor hidden service architecture [3] is comprised of the following components. All Tor onion routers potentially act as *introduction point* and *rendezvous point*.

- *Introduction point* (InP): Tor relays chosen by the hidden service and which are used for forwarding management cells necessary to connect the client and the hidden service at the rendezvous point, and published in the directory server to serve as Bob's front interface.
- *Rendezvous point* (RP): A Tor relay chosen by the client which is used to forward all the data between the client and the hidden service.
- *Hidden service directories* (HSDir): Tor relays which the hidden service publishes its descriptors, and accessed by clients in order to learn the addresses of the hidden service's introduction points.

Figure 4 illustrates the eight steps of the circuit creation procedure of the hidden service. The detail of each step is as follows:

1. Bob (hidden server) randomly picks several ORs as the introduction points, and builds two-hop circuits to them. He sends a *RELAY_COMMAND_ESTABLISH-_INTRO* cell which contains key length, public key, hash of session info and signature of above information to each of these introduction points. Upon receiving this cell, the introduction point checks the information. Once success, the introduction point replies a *RELAY_COMMAND_INTRO_ESTABLISHED* cell with

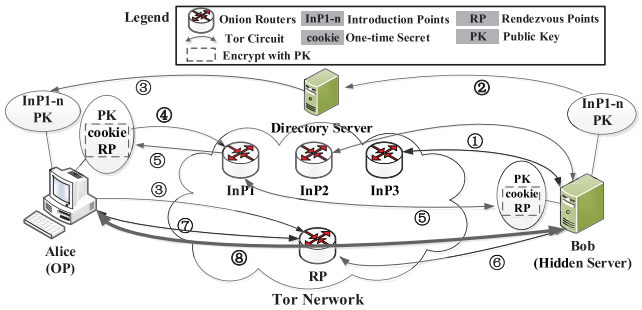

Fig. 4 Structure of hidden service.

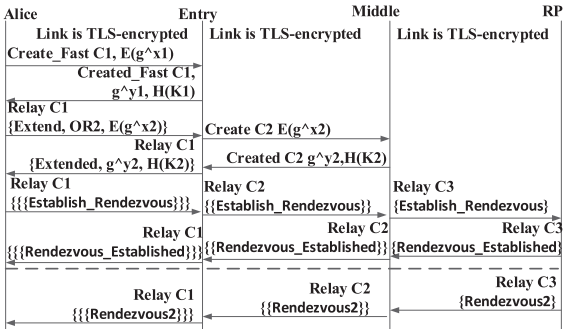

Fig. 5 Client side protocol-feature.

an empty payload to inform Bob that the circuit is established.

2. Bob assembles a hidden service descriptor, containing his public key and a summary of each introduction points, and signs this descriptor with its private key. He uploads the descriptor to a distributed hash table on the Directory Server. Then, Bob can public his descriptors in a public place, and clients can access the hidden service via Tor. After this step, the hidden service is set up.
3. Alice (client) initiates connection establishment by downloading the descriptor from the onion address of Bob, and the onion address with *.onion* postfix can be derived from the service's public key (e.g., *XYZ.onion*, where "*XYZ*" consists of 16 random characters). Then, Alice learns information of the introduction points set and the right public key. Around this time, Alice also builds a circuit to another randomly picked router and requests it to act as the rendezvous point by assigning it a one-time secret. Alice sends a *RELAY_COMMAND_ESTABLISH_RENDEZVOUS* cell to initiate a circuit to the rendezvous point. Once receiving this cell, the rendezvous point replies a *RELAY_COMMAND_RENDEZVOUS_ESTABLISHED* cell to inform Alice that the circuit is established. Figure 5 illustrates the circuit creation procedure, it is a two-hop circuit between Alice and the rendezvous point.
4. Alice assembles an introduce message (encrypted by the hidden service's public key) including the address of the rendezvous point and the one-time secret. Then, Alice creates a three-hop circuit to one of the introduction point, and sends a *RELAY_COMMAND_INTRODUCE1* cell to the introduction point contains the introduce message.
5. Upon receiving the *RELAY_COMMAND_INTRODUCE1* cell, the introduction point repacks the cell into a new *RELAY_COMMAND_INTRODUCE2* cell and send it along the corresponding circuit to Bob (If the *PK_ID* was unrecognized, the *RELAY_COMMAND_INTRODUCE1* cell will be discarded). The introduction point also replies a *RELAY_COMMAND_INTRODUCE_ACK* cell to Alice. Upon receiving this *ACK* cell, Alice tears down the circuit to the introduction point.
6. After Bob receives the *RELAY_COMMAND_INTRODUCE2* cell, he extracts the rendezvous point's nickname, the rendezvous cookie, and the *Diffie-Hellman* value of g^x chosen by Alice with the private key for the corresponding hidden service. Bob generates another half of the *Diffie-Hellman* value g^y and derives the key g^{xy}. Then, Bob establishes a three-hop circuit to the rendezvous point and sends a *RELAY_COMMAND_RENDEZVOUS1* cell to the rendezvous point along the circuit. The cell contains the key data g^y, the hash value of the key $H(g^{xy})$ and the rendezvous cookie.
7. After the rendezvous point receives the *RELAY_COMMAND_RENDEZVOUS1* cell, it checks the rendezvous cookie from Alice and Bob. If the two cookies were matched, the rendezvous point removes the rendezvous cookie and repacks the rest data into a *RELAY_COMMAND_RENDEZVOUS2* cell, and forwards the cell along the rest of corresponding circuit to Alice.
8. When Alice receives the *RELAY_COMMAND_RENDEZVOUS2* cell, she uses g^y to generate the key g^{xy} and does verification by comparing the key with the hash value $H(g^{xy})$ to complete the handshake. The data transmitted between Alice and Bob will be encrypted by this key as the first onion skin. Then, Alice sends the *RELAY_COMMAND_BEGIN* cell to Bob via the six-hop circuit to open the TCP connections. In this process, multiple *RELAY_COMMAND_BEGIN* cells may be sent along the circuit, and multiple streams may be opened between Alice and Bob. However, Alice must not send these cells to any other addresses along the circuit, even if she did, all of these cells will be rejected.

4. Protocol-Feature Attack

In this section, we elaborate our attack for locating the client of hidden service. Entry routers are the first hop of the circuit, and only the entry routers know the real IP address of the client. Without loss of generality we assume that an attacker maintains a malicious website and intends to know who accesses the web site. We will discuss the feasibility of this assumption in the discussion part. Then, the attacker manipulates some onion routers into the Tor network and controls the entry onion router. This is reasonable that onion routers of Tor network are set up in a voluntary man-

ner [13], the same assumption was also made in attacks [5]–[12], [14], [20] against Tor network. In addition, the attacker also need to set up a server that builds TCP connections to the hidden server and the controlled entry routers to record information of related cells and do analysis denoted as central server. We denote circuit ID, source IP, destination IP, timestamp and cell type as related information (*RI*).

4.1 Client Side Protocol-Feature

Figure 3 illustrates the creation process of a normal three-hop Tor circuit. After the three-hop circuit is established, Alice will immediately send the *RELAY_BEGIN* cell to start downloading data. However, in circuit creation procedure of hidden service, after Alice establishes circuit to the rendezvous point, she will enter into listening state and will not send the *RELAY_BEGIN* cell. Until Alice receives the *RELAY_COMMAND_RENDEZVOUS2* cell, which indicating the six-hop hidden circuit between Alice and Bob is established, then Alice will send the *RELAY_BEGIN* cell to the hidden server to start communicating. Detailed procedure is shown in Fig. 5. During this process, we can observe a special combination of cells denoted as the Client Side Protocol-Feature (*CSPF*):

- The entry router receives 1 × *CREATE_FAST_CELL* + 2 × *CELL_RELAY* cells from Alice, and sends 1 × *CREATED_FAST_CELL* + 3 × *CELL_RELAY* cells (include a Rendezvous2 cell) to Alice.
- The entry router receives 1 × *CREATED_CELL* + 3 × *CELL_RELAY* cells (include a Rendezvous2 cell) from middle, and send 1 × *CREATE_CELL*+ 1 × *CELL_RELAY* cell.

After the whole circuit established, Alice will send the *RELAY_BEGIN* cell. Our controlled entry router can detect this cell based on the *CSPF* (the send and receive timing and the counts of each cell must correspond to the relationship shown in Fig. 5), even if the entry router cannot decrypt the cell and obtain the content. In addition, Tor adopts a bandwidth weighted node (router) selection algorithm in order to enhance its performance [13], and the key factors for choosing the entry routers are high bandwidth and long online time. So the CSPF does not have the function choose any router as the entry router. Detailed steps are elaborated in the following section.

4.2 Basic Idea

Figure 6 illustrates the four steps of our basic idea:

1. In Step 5 of Fig. 4, Bob (hidden server) receives the *RELAY_COMMAND_INTRODUCE2* cell from the introduction point, and the cell contains basic information of the rendezvous point. Bob reports *RI* of this cell to the central server to indicate the start of location. Then, Bob sends the *RELAY_COMMAND_RENDEZVOUS1* cell to the rendezvous point to build a circuit.

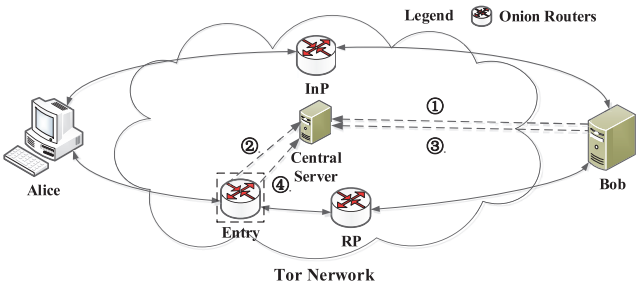

Fig. 6 Communication process.

2. Our controlled routers keep monitoring the traffic flow. If one of the controlled routers was chosen as the entry router, our entry router can observe a combination of cells matching the *CSPF* in Sect. 4.1 during the circuit creation process and report to the central server. However, the *CSPF* dose not necessarily determine that our router is chosen as the entry, we need to confirm this is a true positive by executing the next steps. After the circuit is successfully established between Alice (client) and Bob, Alice starts to send cells contain application data to Bob as shown in Step 8 of Fig. 4. Our entry router manipulates an appropriate cell and sends it to Bob along the circuit.
3. When Bob receives the manipulated cell, Bob cannot correctly decrypt this cell. Bob reports the decryption error caused by the manipulated cell's *RI* to the central server and tear down the circuit by sending the *CELL_DESTROY* cell to Alice along the circuit.
4. When our controlled entry router detect the *CELL_DESTROY* cell, it will report to the central server.

After these four steps, we search for correlation among the time when the entry router sends manipulated cell, the time when Bob sends the destroy cell, and the time when entry router receives the destroy cell. Sice the entry router is first hop of Alice, it knows the real IP address of Alice, so we can locate Alice.

4.3 Detailed Process

Phase 1: Presumably identify the *CSPF* and locate the client

From Step 1 to Step 3 of Fig. 4, Alice builds a circuit to an onion router, and sends the *RELAY_COMMAND_ESTABLISH_RENDEZVOUS* cell to notify the router to run as the rendezvous point. Then, Alice keeps to listen. The introduction point repackages the *RELAY_COMMAND_INTRODUCE1* cell into the *RELAY_COMMAND_INTRODUCE2* cell and sends it to Bob in Step 5 of Fig. 4. When Bob receives this cell, he obtains the rendezvous point's information and will build a circuit to the rendezvous point in Step 6 of Fig. 4. Once the circuit is established, Bob immediately sends the *RELAY_COMMAND_RENDEZVOUS1* cell to the rendezvous point and report *RI* of this cell to the central server. Upon receiving the *RELAY_COMMAND_RENDEZVOUS1*

cell, the rendezvous point repacks the cell into the *RELAY_COMMAND_RENDEZVOUS2* cell and forward the cell along corresponding circuit to Alice.

As shown in Fig. 5, we choose the *CELL_CREATE_FAST* cell as the start-tag of Phase 1 and report this information to the central server. If Alice could receive the *RELAY_COMMAND_RENDEZVOUS2* cell sent by the rendezvous point, she knows a hidden circuit is successfully established. We choose this cell as the finish-tag of Phase 1 and report this information to the central server. During the Phase 1, if our controlled routers relay and receive a combination of cells matching the *CSPF* and report the *RIs* of each cell to the central server, it does not necessarily determine that one of our controlled routers is chosen by Alice, and we need the operations done in Phase 2 to confirm this is a true positive.

Phase 2: Verify the client

After the hidden circuit is established, Alice sends the *RELAY_COMMAND_BEGIN* cell to open a stream with Bob as shown in Step 7 and Step 8 of Fig. 4. We denote this cell as the start-tag of Phase 2. Our entry router can recognize this cell based on the *CSPF*. Our entry router modifies one bit of the cell and reports *RI* of this cell to the central server. Then, our entry router sends this modified cell along the circuit to Bob. As shown in Fig. 7, the circuit of a hidden service is end-to-end encrypted, and integrity checking can only be recognized after all onion layers of encryption are removed, so the onion routers along the circuit which lack of integrity verification cannot detect this modified cell.

After Bob receives the modified *RELAY_COMMAND_BEGIN* cell, he is unable to correctly decrypt the cell. Based on the mechanism of Tor [1], Bob will tear down the circuit between Alice and himself by sending the *CELL_DESTROY* cell. Bob will report *RI* of the modified cell and record the timeing of this cell. When our controlled entry router receives the *CELL_DESTROY* cell, it will report *RI* of this cell to the central server, and we take the *CELL_DESTROY* cell as the finish-tag of Phase 2.

Phase 3: Proof the real identity by time correlation

The central server may receive a large number of cell information after the first two phases. In this phase, we establish appropriate rules to filter out useless information. Herein we discuss in detail about each phase.

- The *CREATE_FAST* cell indicates the start-tag of the Phase1 and the *RELAY_COMMAND_RENDEZVOUS2* cell indicates the finish-tag. When the central server receives the finish-tag of Phase 1, it filters out the useless cell *RIs* by checking the number and the timing of *CREATE*, *CREATED* and *RELAY* cells not matching the *CSPF*. There may remain N circuits: if $N = 0$, none of our controlled routers is chosen as the entry, and the attack is failed. If $N \geq 1$, some controlled routers are chosen as the entry of some hidden circuit, we need to confirm whether one of them is chosen by Alice, and we perform the next steps.
- The modified *RELAY_COMMAND_BEGIN* cell sent by the entry router indicates the start-tag of Phase 2, the central server records the timing of this cell, denoted as T_s. After Bob receives the modified cell, the central server records the timing as T_d. Due to decryption error, Bob will tear down the circuit by send the *CELL_DESTROY* cell. The *CELL_DESTROY* cell received by the entry router indicates the finish-tag of this Phase, denoting the timing as T_f.
- After Phase 2 finished, the central server filters out the circuits which cannot detect the *CELL_DESTROY* cell during Phase 2, and there remain some circuits. Then, the central server finds a circuit contains the three key timestamps T_s, T_d, T_f, and verifying whether the time relationship satisfies the condition $T_s < T_d < T_f$. If such a condition was set up, the central server determines the *CELL_DESTROY* cell is caused by our manipulated cell, and the entry router next to Alice is controlled. So the source IP address of this circuit is the real IP address of Alice.

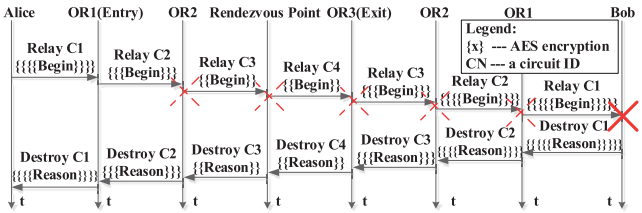

Fig. 7 Modify a cell at the entry router.

5. Analysis

The key factor for our paper is that we can control the entry router [14] of the client. In this section, we analyze the probability that our controlled router is selected by the client as the entry router, and an approach that can increases the catch probability.

5.1 Assign Flag

According to Tor design document [17], the onion routers are divided into four categories, Guard-flagged (entry) nodes [15], non-flagged nodes, Exit-flagged(exit) nodes, Guard+Exit-flagged (either entry or exit, denote as double) nodes.

- Stable: The routers are active and mean upload time is above the median of known active routers or at least 7 days. Routers are never assigned stable flag if they are running the alpha version Tor client.
- Guard: The routers are stable, and can provide bandwidth above a certain threshold (the median of all routers current bandwidths, or 250 KB/s). These routers are selected to receive the Guard flag, marked as the entry router.
- Exit: The routers are stable, and allow exits at least two of the ports 80, 443, and 6667 and allows exist at least

one /8 address space.

- Guard+Exit (Double): The routers which both receive the entry and exit flag are chosen as guard+exit routers, and these routers provide bandwidth both to the entry routers and the exit routers. We denote this type of routers as double routers.

5.2 Catch Probability Analysis

Tor currently uses bandwidth weighted router selection algorithm, details are illustrated in the official design document [13], [17]. We denote the bandwidth set of all routers as $B = \{B_1, B_2, \ldots, B_n, B_{n+1}, \ldots, B_{n+k+1}\}$. Denote bandwidth of entry routers, exit routers, double routers as B_g, B_e, B_d. We use the i^{th} router's bandwidth to calculate the probability of the i^{th} router is chosen as the entry router of the circuit as

$$P(i) = B_i / B_G \tag{1}$$

$$W_{gd} = \begin{cases} 1 - \frac{B}{3(B_g + B_d)} : W_{gd} \geq 0 \\ 0 : W_{gd} < 0 \end{cases} \tag{2}$$

$$W_{ed} = \begin{cases} 1 - \frac{B}{3(B_e + B_d)} : W_{ed} \geq 0 \\ 0 : W_{ed} < 0 \end{cases} \tag{3}$$

$$B_G = B_g + B_d \cdot W_{ed} \tag{4}$$

Where W_{gd} is the bandwidth weight of entry routers in double routers, W_{ed} is the exit weight, B_g is the total bandwidth for pure entry nodes. If $B_g < (B/3)$, $W_{gd} < 0$ the entry bandwidth is scarce, the entry will not be considered for nonentry use. Then the entry bandwidth is calculated as Eq. (4), Tor uses double routers bandwidth to supply the entry routers bandwidth. The exit bandwidth policy is similar to the entry bandwidth.

After we manipulate a set of k controlled routers $B_c = \{B_{n+1}, \ldots, B_{n+k+1}\}$ into Tor network. We assume these routers advertise same bandwidth, for example, $B_{n+1} = B_{n+2} = \ldots = B_{n+k+1} = b$. Then, the i^{th} router's catch probability can be calculated as

$$P(k, B_i) = B_c / B'_G \; (B_c = \sum_{n+1}^{n+k+1} B_i = k \cdot b) \tag{5}$$

$$B'_G = B'_g + B'_d = (B_g + B_c) + B_d \cdot W'_{ed} \tag{6}$$

$$W'_{ed} = 1 - \frac{B + k \cdot b}{3(B_e + B_d)} \tag{7}$$

According to Eq. (5), the catch probability $P(k, B_i)$ is depends on both bandwidth and number of controlled routers.

We did simulation analysis experiment in MATLAB, and Fig. 8 illustrates the relationship between the catch probability and the bandwidth, as well as the number of entry routers. From this figure, if we manipulated 800 entry routers of 30MB/s bandwidth in Tor network, the theoretical catch probability will approach about 96%. The current Tor network has 1947 pure entry routers and 447 double routers (we collected the data from the Tor official web site [24] on May 13, 2014), and the percentage of the controlled entry routers in the entire Tor entry router is around 800/(1947 + 447 + 800) ≈ 25%.

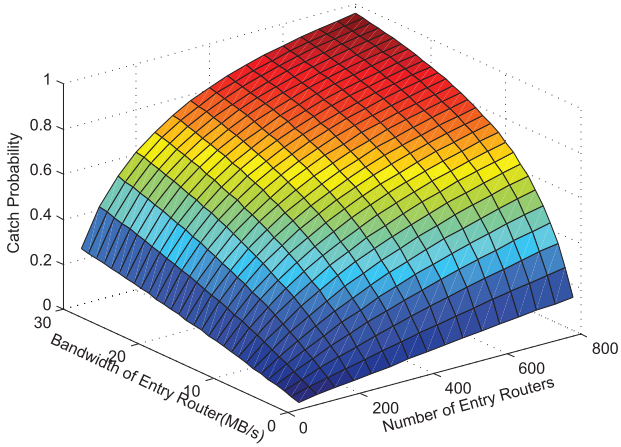

Fig. 8 Catch Probability vs. Bandwidth and Number of Entry Routers.

5.3 Improve the Catch Probability

According to Eq. (5), we derive the following condition:

$$P(k, B_i) \geq P(t, B_j), if k \cdot B_i \geq t \cdot B_j \tag{8}$$

Based on Eq. (8), if the catch probability $P(k, B_i)$ was a monotonous increasing function in terms of number k and bandwidth of controlled entry routers b respectively, so it can be determined by the total bandwidth of the controlled entry routers set $B_c = k \cdot b$. We proofed this equation in Appendix. So we can increase total bandwidth of our controlled routers to improve the catch probability.

5.4 Performance Matric

We use the correlation coefficient r matric to measure the strength of correlation between the time of generating modified cell and the time of detecting destroy cell. Correlation coefficient is defined by

$$r = \frac{\sum\limits_{x,y} (x - \overline{x})(y - \overline{y})}{\sqrt{\sum\limits_{x} (x - \overline{x})^2} \sqrt{\sum\limits_{y} (y - \overline{y})^2}} \tag{9}$$

where x is the time of generating modified cell, y is the time of detecting destroy cell. $\overline{x}$ and $\overline{y}$ are the mean values of x and y.

6. Evaluation

We have implemented the proposed attack in Sect. 4, and

the experimental results have proved the theoretical analysis presented in Sect. 5.

6.1 Experiment Setup

Our experiment setting is shown in Fig. 9. We use Firefox 24.4.0 running on CentOS 6.5 to access the hidden web site which is deployed on the Apache server in version 2.2.15 on CentOS 6.5 to offer hidden service, all the onion routers are installed the stable Tor version 0.2.4.21. A MySQL database is deployed on the central server to help to save and analyze data. We modify the torrc file (Tor configuration file) and set the parameters EntryNodes and StricNodes, so the client will choose our controlled entry onion router. We also revise the source code of Tor in order to upload related experimental data to the central server. We deploy the entry router, client and hidden server on three different servers in Japan, Hong Kong and American respectively.

6.2 Experiment Results

To verify the detection rate, we repeat the experiments 500 times for deriving the true positive rate (the probability that the client is accurately located). We also do 500 times experiments that the client does not choose our entry router to evaluate the false positive rate. The true positive rate is 100%, while the false positive rate is 0%.

We have made statistics about Tor routers information based on the official data [24]. By May 13, 2014, there are 4693 onion routers in the Tor network, including 1947 entry routers, 885 exit routers, 447 double routers and 1414 non routers. Figure 10 and Fig. 11 illustrates the bandwidth distribution of collected onion routers. The bandwidth of about 80% routers is lower than 1MB/s. Although this ratio has been improved compared to the same period last year which ratio is more than 90%, onion routers with high bandwidth is still lack in the Tor network. Based on our theoretical analysis presented in Sect. 5, if we could manipulate routers with high bandwidth into Tor network, the probability that our control routers selected as the entry routers will be much higher.

We manipulate 20 entry routers into the Tor network and configure the bandwidth of each router as 10 MB/s in

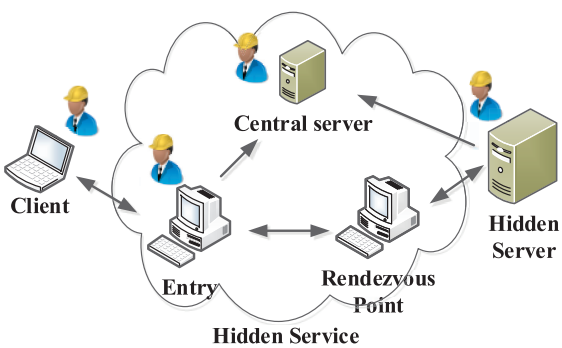

Fig. 9 Experimental network setup.

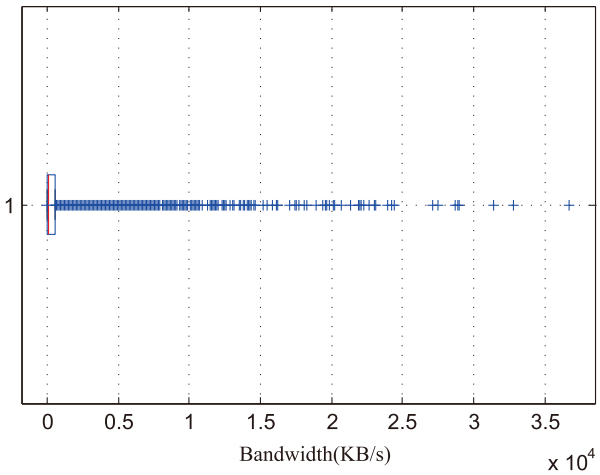

Fig. 10 Box-and-Whisker Diagram.

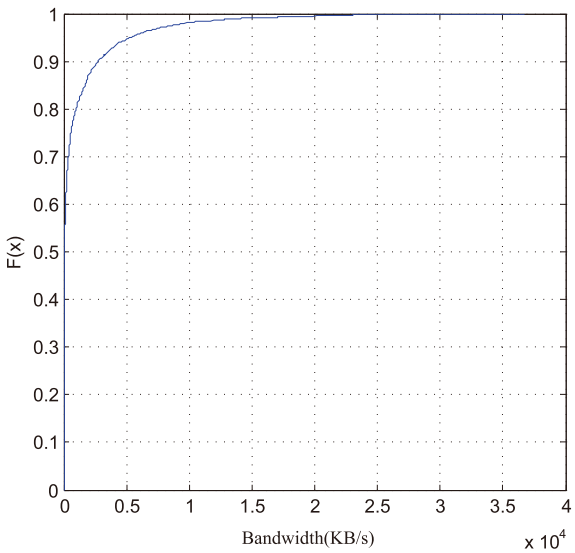

Fig. 11 Cumulative Probability Function.

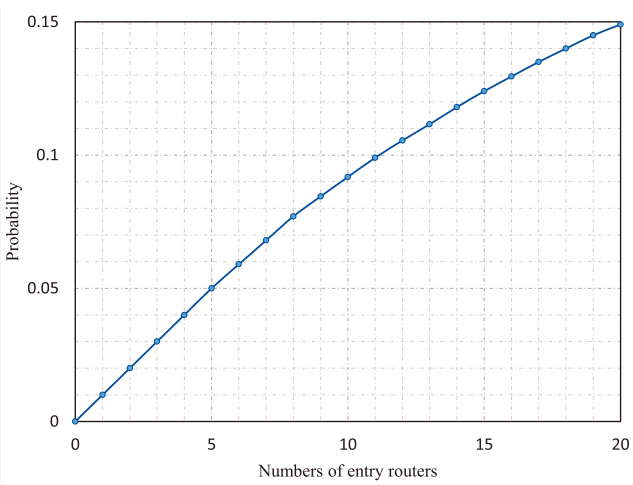

Fig. 12 Correlation between probability and bandwidth of controlled entry routers.

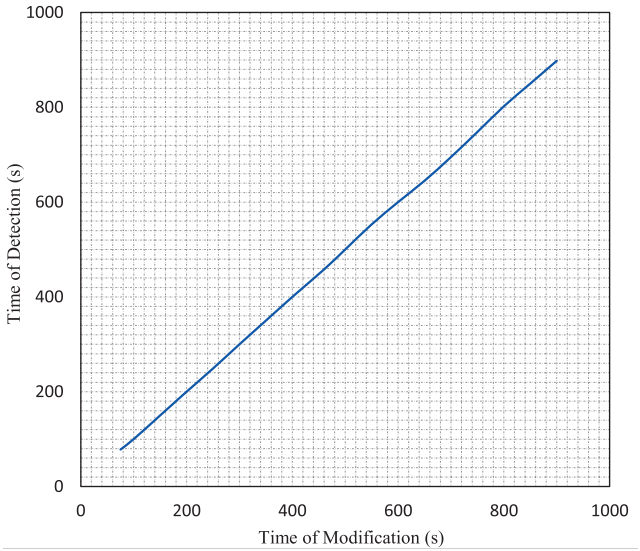

Fig. 13 Correlation between time of modifying cells and Time of detecting decryption error.

the torrc file. Figure 12 illustrates the catch probability is proportional to the increase of the controlled routers.

Figure 13 illustrates the correlation between the time of modifying cells and the time of decryption errors. As we can see, there is a linear correlation, since the actual correlation coefficient between them is one. This strongly confirms that our attack can indeed deanonymize the hidden service and locate the client. In addition to its accuracy, our attack can be very simple, since it only needs to modify and recognize a single cell.

7. Discussion

7.1 Controlled Routers Not Chosen as Entry Router

The key factor for our work is one of our controlled routers chosen as the entry router of the client. We have analyzed and confirmed that the catch probability is in proportion to the number and bandwidth of controlled entry routers. However, based on the path selection mechanism of Tor [13], each Tor node holds a list of trust entry routers, and tends to select its entry router from this list to reject the poor efficiency and malicious nodes [16]. To locate the client or hidden server, firstly we need to reveal the entry guards. A long time online client or server is feasible to locate since we have sufficient time to control the entry routers.

As Fig. 5 illustrates, the middle router will receive one *CELL_CREATE* cell and two *CELL_RELAY* cells, and will relay one *CELL_CREATED* cell and two *CELL_RELAY* cells including a *RELAY_COMMAND_RENDEZVOUS2* cell to the client. If our controlled routers is selected as middle router, we can verify the location of our controlled middle router based on protocol of hidden service. Then, we can reveal the IP address of the suspect entry router or bridge. After we locate the entry router of the client, we can control of this router. Then we can apply our attack to locate the client.

7.2 Feasibility to Control the Hidden Server

At the beginning of Sect. 4, we made the assumption that the attacker maintained a hidden server and intended to find out who accesses the web site. This assumption falls within the Tor threat model and without losing of generality. Here we consider the complicated situation that the attacker is not the hidden server's manager. As in references [9], [14], [20], an attacker not only need to control the entry router but also control the exit router of the same circuit. For example, Ling et al. made a stronger assumption in study [9] that the attacker controlled both the entry router and the exit router in the same Tor circuit, and based on another reference [27], the real world catch probability can confirm over 60% by injecting around 4% of onion routers with long uptime and high bandwidth. Compare with them, our attack is easier to implement that the attacker just need to control the end edge of the hidden service (the hidden server or a node between the server side entry router and the hidden server) to monitor the traffic and report the information of the *Destroy* cell. In practical, there are some feasible measures:

- Attackers can put phishing onion addresses in the Internet, which attract users to access their disguise servers.
- Internet Service Providers (ISPs) or network administrators located between the exit router and the hidden server can monitor the network traffic, and attackers can also manipulate nodes between the hidden server and the entry router.
- Attackers can compromise the hidden server or an intermediate node.

Based on the above basis, our assumption is realistic and does no harm the impact of the attack.

7.3 Future Work

There are still some areas for investigation in our work in the future. First, we can investigate the procedure that the client builds circuit to the introduction point to send the rendezvous points' cookie to the hidden server. If we could find a protocol feature, we may improve the true positive and efficiency of the attack. Second, there are some open platforms such as ExperimenTor [18] or Shadow [26], we plan to study the use of these platforms, and deploying our attack in these Tor network testbed.

8. Conclusion

In this paper, we presented a novel protocol-feature attack against Tor's hidden service. Our attack do not rely on traffic analysis or watermarking techniques, and we aim directly at the protocol of hidden service. Based on the protocol-feature at the client side of hidden service, we studied, designed, implemented, and evaluated our attack. We theoretically analyzed the feasibility of our attack and conducted extensive experiments. The experimental results demonstrated

that we can locate the client of hidden service with high detection rate and low false positive rate.

Acknowledgements

This work is supported by NSFC (Grant Nos. 61300181, 61502044), the Fundamental Research Funds for the Central Universities (Grant No. 2015RC23).

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Proc. of the USENIX Security Sysposium, San Diego, CA, USA, pp.303–320, 2004.
[2] The Tor Project, Inc., "Tor: Anonymity online," https://www.torproject.org/, accessed 2014.
[3] The Tor Project, Inc., "Tor rendezvous specification," https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=re- nd-spec.txt, accessed 2014.
[4] S.J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," Proc. of the ACM Conf.on Comput. Commun. Security (CCS), Alexandria, VA, USA, pp.27–36, Nov. 2006.
[5] L. Overlier and P. Syverson, "Locating hidden servers," Proc. of the IEEE Security Privacy Symposium (S&P), Berkeley, CA, USA, pp.98–114, May 2006.
[6] A. Biryukov, I. Pustogarov I, R. Weinmann, "Trawling for Tor hidden services: Detection, measurement, deanonymization," Proc. of the IEEE Security Privacy Symposium (S&P), San Francisco, CA, USA, pp.80–94, May 2013.
[7] Z. Ling, J.Z. Luo, W. Yu, et al, "Protocol-level attacks against Tor," Comput. Networks, vol.57, no.4, pp.869–886, 2013.
[8] Z. Ling, J.Z. Luo, W. Yu, et al, "Protocol-level hidden server discovery," Proc. of the IEEE Int. Conf. on Comput. Commun. (INFOCOM), Turin, Italy, pp.1043–1051, April 2013
[9] Z. Ling, J.Z. Luo, W. Yu, et al, "A new cell-counting-based attack against Tor," IEEE/ACM Trans. Netw. (TON), vol.20, no.4, pp.1245–1261, 2012.
[10] M.V. Barbera, V.P. Kemerlis, V. Pappas, et al, "Cellflood: Attacking Tor onion routers on the cheap," Comput. Security-ESORICS, Springer Berlin Heidelberg, pp.664–681, 2013.
[11] Bauer, Kevin, et al, "Low-resource routing attacks against tor," Proc. of the ACM workshop on Privacy in Electronic Society, Alexandria, VA, USA, pp.11–20, Oct. 2007.
[12] L. Zhang, J.Z. Luo, M, Yang, et al, "Application-level attack against Tor's hidden service." Proc. of the IEEE Int. Conf. on Pervasive Comput. Appl. (ICPCA), Port Elizabeth, South Africa, pp.509–516, Oct. 2011.
[13] The Tor Project, Inc., "Tor Path Specification," https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt, accessed 2014.
[14] R. Pries, W. Yu, X. Fu, et al, "A new replay attack against anonymous communication networks," Proc. of the IEEE Conf. on Commun. (ICC), Beijing, China, pp.1578–1582, May 2008.
[15] M. Akhoondi, C. Yu, H.V. Madhyastha, "LASTor: A low-latency AS-aware Tor client," Proc. of the IEEE Security Privacy Symposium (S&P), San Francisco, CA, USA, pp.476–490, May 2012.
[16] T. Elahi, K. Bauer, M. AlSabah, et al, "Changing of the guards: A framework for understanding and improving entry guard selection in Tor," Proc. of the ACM Workshop on Privacy in Electronic Society, Raleigh, NC, USA, pp.43–54, Oct. 2012.
[17] The Tor Project, Inc., "Tor directory protocol, version 3," https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=di- r-spec.txt, accessed 2014.
[18] K. Bauer, M. Sherr, M. Mccoy, et al, "ExperimenTor: A testbed for safe and realistic Tor experimentation," Proc. of the USENIX Workshop on Cyber Security Experimentation and Test (CSET), pp.51–59, Aug. 2011.
[19] X. Wang and D.S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter-packet delays," Proc. of the ACM Conf.on Comput. Commun. Security (CCS), Washington, DC, USA, pp.20–29, Oct. 2003.
[20] G.F. He, M. Yang, X.D. Gu, et al, "A novel active website fingerprinting attack against Tor anonymous system," Proc. of IEEE Int. Conf. on Comput. Supported Cooperative Work in Design (CSCWD), Hsinchu, Taiwan, pp.112–117, May 2014.
[21] G. Danezis, R. Dingledine, N. Mathewson, "Mixminion: design of a type III anonymous remailer," Proc. of the IEEE Security Privacy Symposium (S&P), Oakland, CA, USA, pp.2–15, May 2003.
[22] K. Loesing K, S.J. Murdoch, R. Dingledine, "A case study on measuring statistical data in the tor anonymity network," Financial Cryptography and Data Security. Springer Berlin Heidelberg, pp.203–215, 2010.
[23] M.H. Mohajeri, B. Li, M. Derakhshani, et al, "SkypeMorph: Protocol obfuscation for Tor bridges," Proc. of the ACM Conf. on Comput. Commun. Security (CCS), Raleigh, NC, USA, pp.97–108, 2012.
[24] The Tor Project, Inc., "Tor network status," https://metrics.torproject-//t.org/network.html, accessed 2014.
[25] J. Zhang, H.X. Duan, W. Liu, et al, "Anonymity analysis of P2P anonymous communication systems," Computer Communications, vol.34, no.3, pp.358–366, 2011.
[26] R. Jansen, N. Hopper, "Shadow: Running Tor in a box for accurate and efficient experimentation," Proc. of the Network and Distributed Security Symposium, Feb. 2012.
[27] X. Fu, Z. Ling, W. Yu, et al, "Network forensics through cloud computing," Proc. of the Int. Workshop on Security and Privacy in Cloud Computing, pp.26–31, June 2010.

Appendix:

We prove Eq. (8) in this appendix.

Condition 1: Let b as a constant parameter and assume k is a continuous variable. The first derivative of function $P(k,b)$ on k is

$$\frac{dP(k,b)}{dk} = \frac{d\left(\frac{k\cdot b}{B'_g+B'_d}\right)}{dk}$$

$$= \frac{b(B'_g+B'_d) - k\cdot b\cdot \frac{d(B'_g+B'_d)}{dk}}{(B'_g+B'_d)^2}$$

$$= \frac{b(B_g + k\cdot b + B_d\cdot(1-\frac{B+k\cdot b}{3(B_e+B_d)}))}{(B'_g+B'_d)^2} - \frac{k\cdot b(b-b\cdot\frac{B_d}{3(B_e+B_d)})}{(B'_g+B'_d)^2}$$

$$= \frac{b(B_g + B_d - \frac{B\cdot B_d}{3(B_e+B_d)})}{(B'_g+B'_d)^2} \qquad (A\cdot 1)$$

If $W'_{ed} = 1 - \frac{B+k\cdot b}{3(B_e+B_d)} > 0$, we can derive $\frac{B}{3(B_e+B_d)} < 1$, so $\frac{dP(k,b)}{dk} > 0$. Else if $W'_{ed} = 0$, we can derive $B'_d = 0$, so $P(k,b) = \frac{k\cdot b}{B_g+k\cdot b} > 0$. So $P(k,b)$ is a monotonous increasing function in terms of k.

Condition 2: According to Condition 1 and Eq. (5), we derive

$$P(k,b) = \frac{k \cdot b}{B_g + k \cdot b + B_d \cdot W'_{ed}(k)} \quad (\text{A}\cdot 2)$$

Let k as a constant parameter and assume b is a continuous variable.

$$P(b,k) = \frac{k \cdot b}{B_g + k \cdot b + B_d \cdot W'_{ed}(b)} \quad (\text{A}\cdot 3)$$

Based on Eq. (A· 2) and (A· 3), we can conclude k and b are exchangeable variables. So use the same procedure in Eq. (8), we can prove $P(b,k)$ is also a monotonous increasing function in terms of b.

Proof: Let $T = k \cdot b$, Eq. (7) can be rewritten

$$P(T) = \frac{T}{B_g + T + B_d \cdot W'_{ed}(T)} \quad (\text{A}\cdot 4)$$

Based on Condition 1 and 2, $P(T)$ is a monotonous increasing function in terms of T.

Rui Wang is a Phd candidate in Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, China. He received the B.S. degree in Computer Science and M.S. degree in Computer Applications from Changchun University of Science and Technology, Changchun, China, in 2009 and 2012. His research interests include anonymous communication, information security and network security. E-mail: ruiwang114@163.com

Qiaoyan Wen received the B.S. and M.S. degrees in Mathematics from Shaanxi normal University, Xi'an, China, in 1981 and 1984, respectively, and the Ph.D degree in cryptography from Xidian University, Xi'an, China, in 1997. She is a professor of Beijing University of Posts and Telecommunications. Her present research interests include coding theory, cryptography, information security, internet security and applied mathematics. E-mail: wqy@bupt.edu.cn

Hua Zhang received the B.S. degree in telecommunications engineering from the Xidian University in 1998, the M.S. degree in cryptology from Xidian University, Xi'an, China in 2005, and the Ph.D degree in cryptology from Beijing University of Posts and Telecommunications in 2008. Now she is a lecturer of Beijing University of Posts and Telecommunications. Her research interests include cryptography, information security and network security. E-mail: zhanghua_288@bupt.edu.cn

Xuelei Li received his B.S. degree in science from Dezhou College in 2009 and M.S. degree in science from University of Jinan, Jinan, Shandong, in 2012. Now he is doing research in Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, China. His interests include the network security and cryptography protocols. E-mail: will8898@163.com