A New Cell-Counting-Based Attack Against Tor

Zhen Ling, Junzhou Luo, *Member, IEEE*, Wei Yu, Xinwen Fu, Dong Xuan, and Weijia Jia

***Abstract*—Various low-latency anonymous communication systems such as Tor and Anonymizer have been designed to provide anonymity service for users. In order to hide the communication of users, most of the anonymity systems pack the application data into equal-sized cells (e.g., 512 B for Tor, a known real-world, circuit-based, low-latency anonymous communication network). Via extensive experiments on Tor, we found that the size of IP packets in the Tor network can be very dynamic because a cell is an application concept and the IP layer may repack cells. Based on this finding, we investigate a new cell-counting-based attack against Tor, which allows the attacker to confirm anonymous communication relationship among users very quickly. In this attack, by marginally varying the number of cells in the target traffic at the malicious exit onion router, the attacker can embed a secret signal into the variation of cell counter of the target traffic. The embedded signal will be carried along with the target traffic and arrive at the malicious entry onion router. Then, an accomplice of the attacker at the malicious entry onion router will detect the embedded signal based on the received cells and confirm the communication relationship among users. We have implemented this attack against Tor, and our experimental data validate its feasibility and effectiveness. There are several unique features of this attack. First, this attack is highly efficient and can confirm very short communication sessions with only tens of cells. Second, this attack is effective, and its detection rate approaches 100% with a very low false positive rate. Third, it is possible to implement the attack in a way that appears to be very difficult for honest participants to detect (e.g., using our hopping-based signal embedding).**

***Index Terms*—Anonymity, cell counting, mix networks, signal, Tor.**

Manuscript received May 29, 2011; accepted November 05, 2011; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Allman. Date of publication January 16, 2012; date of current version August 14, 2012. This work was supported in part by the National Key Basic Research Program of China (973 Program) under Grants 2010CB328104 and 2011CB302800; the National Science Foundation of China (NSFC) under Grants 60903162, 60903161, 61070158, 61070161, 61003257, 61070221, and 61070222/F020802; the US National Science Foundation (NSF) under Grants CNS0916584, CNS1065136, and CNS-1117175; CityU Applied R&D Funding (ARD) under Grants 9681001, 6351006, and 9667052; CityU Strategic Research Grant 7008110; ShenZhen-HK Innovation Cycle Grant ZYB200907080078A; the China Specialized Research Fund for the Doctoral Program of Higher Education under Grant 200802860031; Jiangsu Provincial Natural Science Foundation of China under Grant BK2008030; Jiangsu Provincial Key Laboratory of Network and Information Security under Grant BM2003201; and the Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grant 93K-9. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies. The conference version of this paper was published in the Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), Chicago, IL, November 9–13, 2009.

Z. Ling and J. Luo are with the School of Computer Science and Engineering, Southeast University, Nanjing 210096, China (e-mail: zhenling@seu.edu.cn; jluo@seu.edu.cn).

W. Yu is with the Department of Computer and Information Sciences, Towson University, Towson, MD 21252 USA (e-mail: wyu@towson.edu).

X. Fu is with the Department of Computer Science, University of Massachusetts Lowell, Lowell, MA 01854 (e-mail: xinwenfu@cs.uml.edu).

D. Xuan is with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: xuan@cse.ohio-state.edu).

W. Jia is with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong (e-mail: wei.jia@cityu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2011.2178036

I. INTRODUCTION

CONCERNS about privacy and security have received greater attention with the rapid growth and public acceptance of the Internet, which has been used to create our global E-economy. Anonymity has become a necessary and legitimate aim in many applications, including anonymous Web browsing, location-based services (LBSs), and E-voting. In these applications, encryption alone cannot maintain the anonymity required by participants [1]–[3]. In the past, researchers have developed numerous anonymous communication systems. Generally speaking, mix techniques can be used for either message-based (high-latency) or flow-based (low-latency) anonymity applications. E-mail is a typical message-based anonymity application, which has been thoroughly investigated [4]. Research on flow-based anonymity applications has recently received great attention in order to preserve anonymity in low-latency applications, including Web browsing and peer-to-peer file sharing [5], [6].

To degrade the anonymity service provided by anonymous communication systems, traffic analysis attacks have been studied [3], [7]–[14]. Existing traffic analysis attacks can be categorized into two groups: passive traffic analysis and active watermarking techniques. Passive traffic analysis technique will record the traffic passively and identify the similarity between the sender's outbound traffic and the receiver's inbound traffic based on statistical measures [7]–[9], [15], [16]. Because this type of attack relies on correlating the timings of messages moving through the anonymous system and does not change the traffic characteristics, it is also a passive *timing* attack. For example, Serjantov *et al.* [7] proposed a passive packet-counting scheme to observe the number of packets of a connection that arrives at a mix node and leaves a node. However, they did not elaborate how packet counting could be done. To improve the accuracy of attacks, the active watermarking technique has recently received much attention. The idea of this technique is to actively introduce special signals (or marks) into the sender's outbound traffic with the intention of recognizing the embedded signal at the receiver's inbound traffic [13], [14], [17].

In this paper, we focus on the active watermarking technique, which has been active in the past few years. For example, Yu *et al.* [13] proposed a flow-marking scheme based on the direct sequence spread spectrum (DSSS) technique by utilizing a pseudo-noise (PN) code. By interfering with the rate of a suspect sender's traffic and marginally changing the traffic rate, the attacker can embed a secret spread-spectrum signal into the target traffic. The embedded signal is carried along with the target traffic from the sender to the receiver, so the investigator

can recognize the corresponding communication relationship, tracing the messages despite the use of anonymous networks. However, in order to accurately confirm the anonymous communication relationship of users, the flow-marking scheme needs to embed a signal modulated by a relatively long length of PN code, and also the signal is embedded into the traffic flow rate variation. Houmansadr *et al.* [14] proposed a nonblind network flow watermarking scheme called RAINBOW for stepping stone detection. Their approach records the traffic timing of the incoming flows and correlates them with the outgoing flows. This approach also embeds watermarks into the traffic by actively delaying some packets. The watermark detection problem was formalized as detecting a known spread-spectrum signal with noise caused by network dynamics. Normalized correlation is used as the detection scheme. Their approach can classify a typical SSH connection as a stepping stone connection in 3 min. As we can see, it is hard for the flow-marking technique to deal with the short communication sessions that may only last for a few seconds.

A successful attack against anonymous communication systems relies on accuracy, efficiency, and detectability of active watermarking techniques. Detectability refers to the difficulty of detecting the embedded signal by anyone other than the attackers. Efficiency refers to the quickness of confirming anonymous communication relationships among users. Although accuracy and/or detectability have received great attention [13], [14], [17], to the best of our knowledge, no existing work can meet all these three requirements simultaneously.

In this paper, we investigate a new cell-counting-based attack against Tor, a real-world, circuit-based low-latency anonymous communication network. This attack is a novel variation of the standard timing attack. It can confirm anonymous communication relationship among users accurately and quickly and is difficult to detect. In this attack, the attacker at the malicious exit router detects the data transmitted to a suspicious destination (e.g., server Bob). The attacker then determines whether the data is a *relay* cell or a *control* cell in Tor. After excluding the *control* cells, the attacker manipulates the number of *relay* cells in the circuit queue and flushes out all cells in the circuit queue. In this way, the attacker can embed a signal (a series of "1" or "0" bits) into the variation of the cell count during a short period in the target traffic. An accomplice of the attacker at the entry onion router detects and excludes the *control* cells, records the number of *relay* cells in the circuit queue, and recovers the embedded signal. The signal embedded in the target traffic might be distorted because the cells carrying the different bits (units) of the original signal might be combined or separated at middle onion routers. To address this problem, we develop the recovery algorithms to accurately recognize the embedded signal. Our theoretical analysis shows that the detection rate is a monotonously increasing function with respect to the delay interval and is a monotonously decreasing function of the variance of one way transmission delay along a circuit. In our real-world experiments, the experimental results match the theoretical results well. To be specific, our attack needs only 2 s to achieve a true positive rate of almost 100% and the false positive rate of almost 0%.

We have implemented the cell-counting-based attack against Tor and performed a set of real-world Internet experiments to validate the feasibility and effectiveness of the attack. The attack presented in this paper is one of the first to exploit the implementation of known anonymous communication systems such as Tor by exploiting its fundamental protocol design. There are several unique features for this attack. First, this attack is highly efficient and can quickly confirm very short anonymous communication sessions with tens of cells. Second, this attack is effective, and its detection rate approaches 100% with very low false positive rate. Third, the short and secret signal makes it difficult for others to detect the presence of the embedded signal. Our time-hopping-based signal embedding technique makes the attack even harder to detect. The attack poses a significant threat to the anonymity provided by Tor because the attack can confirm over half of communication sessions by injecting around 10% malicious onion routes on Tor [18], [19].

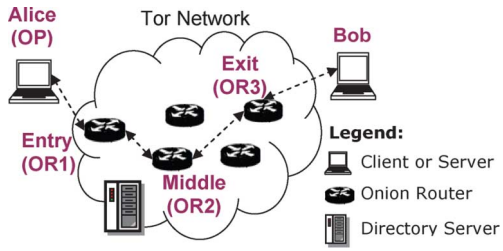


Fig. 1. Tor network.

The remainder of this paper is organized as follows: We introduce the background in Section II. We present the cell-counting-based attack, including the basic idea, issues of the attack, and solutions, in Section III. In Section IV, we discuss various issues, including some extension, and the detectability and impact of the proposed attack. In Section V, we analyze the effectiveness of the attack. In Section VI, we show experimental results on Tor and validate our findings. We review related work in Section VII and conclude this paper in Section VIII.

II. Background

In this section, we first overview the components of Tor. We then present the procedures of how to create circuits and transmit data in Tor and process cells at onion routers.

A. Components of Tor

Tor is a popular overlay network for providing anonymous communication over the Internet. It is an open-source project and provides anonymity service for TCP applications [20]. As shown in Fig. 1, there are four basic components in Tor.

1) *Alice* (i.e., *Client*): The client runs a local software called *onion proxy* (*OP*) to anonymize the client data into Tor.
2) *Bob* (i.e., *Server*): It runs TCP applications such as a Web service.
3) *Onion routers (ORs)*: Onion routers are special proxies that relay the application data between Alice and Bob. In Tor, transport-layer security (TLS) connections are used for the overlay link encryption between two onion routers. The application data is packed into equal-sized cells (512 B as shown in Fig. 2) carried through TLS connections.
4) *Directory servers*: They hold onion router information such as public keys for onion routers. Directory authorities hold authoritative information on onion routers, and directory caches download directory information of onion routers from authorities. A list of directory authorities is

Fig. 2. Cell format by Tor. (a) Tor cell format. (b) Tor relay cell format.

hard-coded into the Tor source code for a client to download the information of onion routers and build circuits through the Tor network.

Fig. 2 illustrates the cell format used by Tor. All cells have a 3-B header, which is not encrypted in the onion-like fashion so that the intermediate Tor routers can see this header. The other 509 B are encrypted in the onion-like fashion. There are two types of cells: *control* cell shown in Fig. 2(a) and *relay* cell shown in Fig. 2(b). The command field (*Command*) of a control cell can be: *CELL_PADDING*, used for keepalive and optionally usable for link padding, although not used currently; *CELL_CREATE* or *CELL_CREATED*, used for setting up a new circuit; and *CELL_DESTROY*, used for releasing a circuit. The command field (*Command*) of a relay cell is *CELL_RELAY*. Note that relay cells are used to carry TCP stream data from Alice to Bob. The relay cell has an additional header, namely the relay header. There are numerous types of relay commands (*Relay Command*), including *RELAY_COMMAND_BEGIN*, *RELAY_COMMAND_DATA*, *RELAY_COMMAND_END*, *RELAY_COMMAND_SENDME*, *RELAY_COMMAND_EXTEND*, *RELAY_COMMAND_DROP*, and *RELAY_COMMAND_RESOLVE*. Note that all these can be found in *or.h* in released source code package by Tor.

B. Circuit Creation and Data Transmission

In Tor, an *OR* maintains a TLS connection to other *OR*s or *OP*s on demand. The *OP* uses a way of source routing and chooses several *OR*s (preferably ones with high bandwidth and high uptime) from the locally cached directory, downloaded from the directory caches. The number of the selected *OR*s is referred as the path length. We use the default path length of three as an example. The *OP* iteratively establishes circuits across the Tor network and negotiates a symmetric key with each *OR*, one hop at a time, as well as handles the TCP streams from client applications. The *OR* on the other side of the circuit connects to the requested destinations and relays the data.

We now illustrate the procedure that the *OP* establishes a circuit and downloads a file from the server. *OP* first sets up a TLS connection with *OR1* using the TLS protocol. Then, tunneling through this connection, *OP* sends a *CELL_CREATE* cell and uses the *Diffie–Hellman* (DH) handshake protocol to negotiate a base key $K_1 = g^{xy}$ with *OR1*, which responds with a *CELL_CREATED* cell. From this base key material, a forward symmetric key kf_1 and a backward symmetric key kb_1 are produced [21]. In this way, a 1-hop circuit *C1* is created. Similarly, *OP* extends the circuit to a 2-hop circuit and 3-hop circuit. After the circuit is set up between the *OP* and *OR3*, *OP* sends a *RELAY_COMMAND_BEGIN* cell to the exit onion router, and the cell is encrypted as $\{\{\{Begin < IP, Port >\}_{kf_3}\}_{kf_2}\}_{kf_1}$, where the subscript refers to the key used for encryption of one onion skin. The three layers of onion skin are removed one by one each time the cell traverses an onion router through the circuit. When *OR3* removes the last onion skin by decryption, it recognizes that the request intends to open a TCP stream to a *port* at the destination *IP*, which belongs to Bob. Therefore, *OR3* acts as a proxy, sets up a TCP connection with Bob, and sends a *RELAY_COMMAND_CONNECTED* cell back to Alice's *OP*. Then, Alice can download the file.

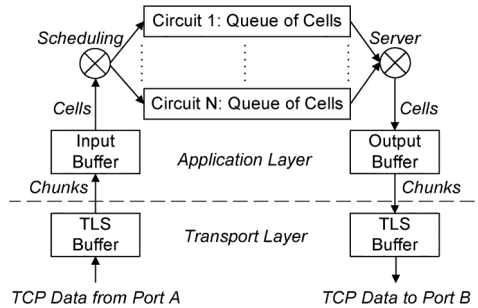


Fig. 3. Processing the cells at onion routers.

C. Processing Cells at Onion Routers

Fig. 3 illustrates the procedure of processing cells at onion routers. Note that the cells mentioned below are all *CELL_RELAY_DATA* cells, which are used to carry end-to-end stream data between Alice and Bob. To begin with, the onion router receives the TCP data from the connection on the given port *A*. After the data is processed by TCP and TLS protocols, the data will be delivered into the TLS buffer of the connection. When there is pending data in the TLS buffer, the read event of this connection will be called to read and process the data. The connection read event will pull the data from the TLS buffer into the connection input buffer. Each connection input buffer is implemented as a linked list with small chunks. The data is fetched from the head of the list and added to the tail. After the data in the TLS buffer is pulled into the connection input buffer, the connection read event will process the cells from the connection input buffer one by one. As stated earlier, the cell size is 512 B. Thus, 512-B data will be pulled out from the input buffer every time until the data remaining in the connection input buffer is smaller than 512 B. Since each onion router has a routing table that maintains the map from source connection and circuit ID to destination connection and circuit ID, the read event can determine that the transmission direction of the cell is either in the forward or backward direction. Then, the corresponding symmetric key is used to decrypt/encrypt the payload of the cell, replace the present circuit ID with the destination circuit ID, and append the cell to the destination circuit queue. If it is the first cell added to this circuit queue, the circuit will be made active by being added into a double-linked ring of circuits with queued cells waiting for a room to free up on the output buffer of the destination connection. Then, if there is no data waiting in the output buffer for the destination connection, the cell will be written into the output buffer directly, and then the write event of this circuit is added to the event queue. Subsequent incoming cells are queued in the circuit queue.

When the write event of the circuit is called, the data in the output buffer is flushed to the TLS buffer of the destination connection. Then, the write event will pull as many cells as possible from the circuit queue of the currently active circuit to the output

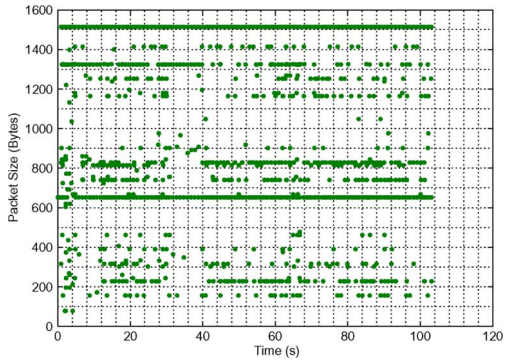


Fig. 4. Packet sequence versus packet size.

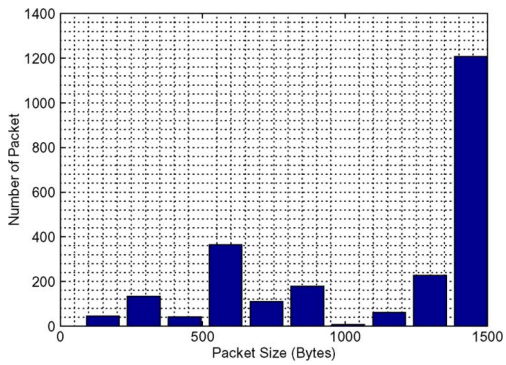


Fig. 5. Number of packets versus packet size.

buffer and add the write event of this circuit to the event queue. The next write event can carry on flushing data to the output buffer and pull the cells to the output buffer. In other words, the cells queued in the circuit queue can be delivered to the network via port *B* by calling the write event twice.

III. Cell-Counting-Based Attack

In this section, we first show that the size of IP packets in the Tor network is very dynamic. Based on this finding, we then introduce the basic idea of the cell-counting-based attack and list some challenging issues related to the attack and present solutions to resolve those issues.

A. Dynamic IP Packet Size of Traffic Over Tor

In Tor, the application data will be packed into equal-sized cells (e.g., 512 B). Nonetheless, via extensive experiments over the Tor network, we found that the size of IP packets transmitted over Tor is dynamic. Fig. 4 shows the size of received IP packets at the client over time, and Fig. 5 shows the frequency of the IP packet size. It can be observed that the size of packets from the sender to the receiver is random over time, and a large number of packets have varied sizes, other than the cell size or maximum transmission unit (MTU) size.

These observations can be reasoned as follows.

1) The varied performance of onion routers may cause cells not to be promptly processed. According to cell processing in Fig. 3, if an onion router is overloaded, unprocessed cells will be queued. Therefore, cells will be merged at the IP layer and sent out together. Those merged cells may be split into multiple MTU-sized packets and one non-MTU-sized packet.

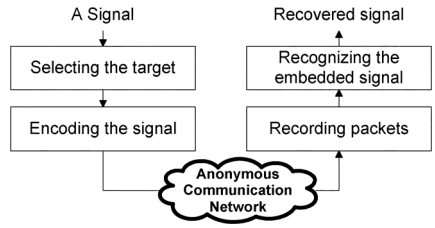


Fig. 6. Cell-counting-based attack.

2) Tor network dynamics may incur those non-MTU-sized IP packets as well. If the network between onion routers is congested, cells will not be delivered on time. When this happens, cells will merge, and non-MTU-sized IP packets will show up.

B. Basic Idea of Cell-Counting-Based Attack

As we stated above, the packet size observed at the client shows a high probability to be random because of the performance of onion routers and Internet traffic dynamics. Motivated by this finding, we investigate a new cell-counting-based attack against Tor, which allows the attacker to confirm anonymous communication relationship among users very quickly. In addition, it will be hard for the client to detect our developed attack described in what follows.

As we mentioned before, this attack intends to confirm that Alice (client) communicates with Bob (server) over Tor. In order to do so, we assume that the attacker controls a small percentage of exit and entry onion routers by donating computers to Tor. This assumption is also used in other studies [3], [10], [18], [19]. The assumption is valid since Tor is operated in a voluntary manner [21]. For example, attackers may purchase Amazon EC2 virtual machines, which can be put into Tor. The attack can be initiated at either the malicious entry onion router or exit onion router, up to the interest of the attacker. In the rest of the paper, we assume that the attack is initiated at an exit onion router connected to server Bob and intends to confirm that Alice communicates with a known server Bob.

The basic idea is as follows. An attacker at the exit onion router first selects the target traffic flow between Alice and Bob. The attacker then selects a random signal (e.g., a sequence of binary bits), chooses an appropriate time, and changes the cell count of target traffic based on the selected random signal. In this way, the attacker is able to embed a signal into the target traffic from Bob. The signal will be carried along with the target traffic to the entry onion router connecting to Alice. An accomplice of the attacker at the entry onion router will record the variation of the received cells and recognize the embedded signal. If the same pattern of the signal is recognized, the attacker confirms the communication relationship between Alice and Bob.

As shown in Fig. 6, the workflow of the cell-counting-based attack is illustrated as follows.

Step 1: Selecting the Target: At a malicious exit onion router connected to the server Bob, the attacker will log the information, including server Bob's host IP address and port used for a given circuit, as well as the circuit ID. The attacker uses *CELL_RELAY_DATA* cells since those cells transmit the data stream. According to the description of Tor in Section II, we know that the attacker is able to obtain the

first cell backward to the client, which is a *CELL_CREATED* cell and is used to negotiate a symmetric key with the middle onion router. The second cell backward to the client will be a *CELL_RELAY_CONNECTED* cell. All sequential cells will be *CELL_RELAY_DATA* cell, and the attacker starts the encoding process shown in Step 2.

Step 2: Encoding the Signal: In Section II, we introduced the procedure of processing cells at the onion routers. The *CELL_RELAY_DATA* cells will be waiting in the circuit queue of the onion router until the write event is called. Then, the cells in the circuit queue are all flushed into the output buffer. Hence, the attacker can benefit from this and manipulate the number of cells flushed to the output buffer all together. In this way, the attacker can embed a secret signal (a sequence of binary bits, i.e., "10101") into the variation of the cell count during a short period in the target traffic. Particularly, in order to encode bit "1," the attacker flushes three cells from the circuit queue. In order to encode bit "0," the attacker flushes only one cell from the circuit queue. In order to accurately manipulate the number of the cells to be flushed, the attacker needs to count the number of cells in the circuit queue. Once the number of the cells is adequate (i.e,, three cells for encoding "1" bit of the signal, and one cell for "0" bit of the signal), the attacker calls the circuit write event promptly and all the cells are flushed to the output buffer immediately. Unfortunately, due to the network congestion and delay, the cells may be combined or separated at the middle onion routers, or the network link between the onion routers. We will develop a reliable encoding mechanism to deal with network dynamics in Section III-C.

Step 3: Recording Packets: After the signal is embedded in the target traffic in Step 2, it will be transmitted to the entry onion router along with the target traffic. An accomplice of the attacker at the entry onion router will record the received cells and related information, including Alice's host IP address and port used for a given circuit, as well as the circuit ID. Since the signal is embedded in the variation of the cell count for *CELL_RELAY_DATA* cells, an accomplice of the attacker at the entry onion router needs to determine whether the received cells are *CELL_RELAY_DATA* cells. This can be done through a way similar to the one in Step 1. We know that the first two cells that arrive at the entry onion router are *CELL_RELAY_EXTENDED* cells, and the third one is a *CELL_RELAY_CONNECTED* cell. After these three cells, all cells are a *CELL_RELAY_DATA* cell. Therefore, starting from this point, the attacker records the cells arriving at the circuit queue.

Step 4: Recognizing the Embedded Signal: With recorded cells, the attacker enters the phase of recognizing the embedded signal. In order to do so, the attacker uses our developed recovery mechanisms presented in Section III-C to decode the embedded signal. Once the original signal is identified, the entry onion router knows Alice's host IP address, and the exit onion router knows Bob's host IP address of the TCP stream. Therefore, the attacker can link the communication relationship between Alice and Bob. As mentioned earlier, when the signal is transmitted through Tor, it will be distorted because of network delay and congestion. For example, when the chunks of three cells for encoding bit "1" arrive at the middle onion router, the first cell will be flushed to the output buffer promptly if there is no data in the output buffer. The subsequent two cells are queued in the circuit queue. When the write event is called, the first cell is sent to the network, while the subsequent two cells are flushed into the output buffer. Therefore, the chunks of the three cells for carrying bit "1" may be split into two portions. The first portion contains the first cell, and the second portion contains the second and third cell together. Therefore, attention must be paid to take these into account to recognize a signal bit. Due to the network congestion and delay, the cells may be combined or separated at the middle onion routers, or the network link between the onion routers [22]. All these facts cause a distorted version of the originally embedded signal to be received at the entry onion router. To deal with these issues, we will design mechanisms to carefully encode and robustly recover the embedded signal in Section III-C.

C. Issues and Solutions

From the description above, we know that there are two critical issues related to the attack: 1) How can an attacker effectively encode the signal at the exit onion router? 2) How can an attacker accurately decode the embedded signal at the entry onion router? We address these two issues below.

1) Encoding Signals at Exit Onion Routers: Two Cells for Encoding "1" Bit Is Not Enough: As we stated earlier, this attack intends to manipulate the number of cells and embed the secret signal into the variation of the cell count during a short period in the target traffic. If the attacker uses two cells to encode bit "1," it will be easily distorted over the network and will be hard to recover. The reason is that when the two cells arrive at the input buffer at the middle onion router, the first cell will be pulled into the circuit queue. If the output buffer is empty, the first cell will be flushed into the output buffer immediately. Then, the second cell will be pulled to the circuit queue. Since the output buffer is not empty, the second cell will stay in the circuit queue. When the write event is called, the first cell will be delivered to the network, while the second cell will be written to the output buffer and wait for next write event. Consequently, two originally combined cells will be split into two separate cells at the middle router. Hence, the attacker at the entry onion router will observe two separate cells arriving at the circuit queue. These two cells will be decoded as two "0" bits, leading to a wrong detection of the signal. To deal with this problem, the attacker should choose at least three cells for carrying bit "1." If the middle onion router splits them into one cell and two cells, the attacker can still recognize the pattern and decode the signal bit correctly at the entry onion router.

Proper Delay Interval Should Be Selected for Transmitting Cells: Since the signal modulates the number of cells transmitted from the exit onion router to the entry onion router, the delay intervals among cells that carry different units (bits) of the signal will have impact on the accuracy and detectability of the attack. Hence, care must be taken to select a proper interval for transmitting those cells. If the delay interval among cells is too large, users may not be able to tolerate the slow traffic rate and will choose another circuit to transmit the data. When this happens, the attack will fail. When the delay interval among cells is too small, it will increase the chance that cells may be combined at middle onion routers. Let us use one simple example to clarify this. We assume that the delay intervals for three bits "0," "1," and "0" of the signal are very small. The first cell for carrying the first bit "0" arrives at the middle onion router and is written into the queue. This first cell will be flushed into the

output buffer if the output buffer is empty. The write event is added to the event queue, and the cell waits to be written to the network by the write event. Since the interval is small, the three cells for the second bit "1" and the cell for the third bit "0" also arrive at the middle onion router and stay in the circuit queue. When the write event is called, the first cell for carrying the first bit "0" will be written to the network, while the following three cells for carrying the second bit of the signal and one cell for carrying the third bit of the signal will be written to the output buffer all together. When this happens, the original signal will be distorted (i.e., the third bit "0" of the signal will be lost). Therefore, the attacker needs to choose the proper delay interval for transmitting cells. In addition, we will discuss the types of the division and combination of the cells with details in Section III-C.2.

We now check conditions that preserve units of the signal during transmission. Let $S = \{S_0, S_1, \ldots, S_{n-1}\}$ be the signal, a series of bits, where n is the signal length and S_j ($j \in [0, n-1)$]) is 0 or 1. When $S_j = 1$, the attacker will choose three cells to encode bit "1." When $S_j = 0$, the attacker will choose only one cell to encode bit "0." Let the time sequence of the signal S that arrives at the *OR2* be $T = \{T_0, T_1, \ldots, T_{n-1}\}$, and let T_{read} be the average time of calling the read event, which pulls the data of cells for each unit of the signal from the TLS buffer and write them to the circuit queue. Let T_{write} be the average time of calling the write event, which writes the cells in the output buffer to the network and flushes the cells in the circuit queue to the output buffer. Let the delay interval between two sequential bits of the signal be I, and let the delay of transmitting data between *OR3* and *OR2* be D. The relationship between T_i and T_{i+1} can be represented as follows:

$$T_{i+1} = T_i + I + D (0 \le i < n-1). \tag{1}$$

Let the time of the cells for the signal S arriving at the circuit queue be T^{queue}, where $T_i^{\text{queue}} = T_i + T_{\text{read}}$. Let the time of the cells for the signal S arriving at the output buffer be $T^{\text{outbuffer}}$, where $T_i^{\text{outbuffer}} = T_i + T_{\text{read}} + T_{\text{write}}$. Please refer to [22] for statistics of T_{read}, T_{write} and other related random variables.

In order to avoid the combination of cells that belong to different units of a signal in the circuit queue, the cells for carrying one bit should be flushed to the output buffer or the network before the cells for carrying the next unit of the signal arrives at the circuit queue. Therefore, we have

$$T_i^{\text{outbuffer}} \le T_{i+1}^{\text{queue}} \tag{2}$$

$$T_i + T_{\text{read}} + T_{\text{write}} \le T_{i+1} + T_{\text{read}} \tag{3}$$

$$T_i + T_{\text{write}} \le T_i + I + D \tag{4}$$

$$T_{\text{write}} \le I + D. \tag{5}$$

The parameter T_{write} is affected by the network condition. Suppose that the network is congested, i.e., $T_{\text{write}} > I + D$, the write event in the event queue cannot be called in time to flush the cells in the output buffer and the circuit queue. Then, the subsequent cells will be queued in the circuit queue along with the previous cells. Therefore, the cells belonging to different units of the signal will be combined in the circuit queue. If the network load is light and T_{write} is small, i.e., $T_{\text{write}} < I + D$, the cells will be transmitted in time at the middle onion router. In this case, when three cells carrying "1" bit of the signal arrive at the middle onion router, the first cell will be flushed to the output buffer since the output buffer is empty. Then, the next two cells will be queued in the circuit queue. Therefore, the cells for "1" bit of signal will be divided into two parts. If the network load is medium, i.e., $T_{\text{write}} \approx I + D$, when the cells for the previous unit of the signal wait in the output buffer, the cells for the next unit of the signal arrive at the queue. The write event will be called to write the cells for the previous unit of the signal to the network and flush the cells for the next unit of the signal to the output buffer. Therefore, cells for different units of the signal will not be combined or divided.

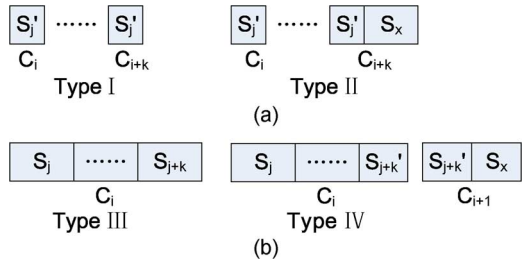


Fig. 7. Signal division and combination. (a) Types I and II. (b) Types III and IV.

2) Decoding Signals at Entry Onion Routers: Distortion of the Signal: The proper selection of delay interval for transmitting cells for carrying different units of the signal will reduce the probability that cells will be combined or divided at middle onion routers. However, due to unpredictable network delay and congestion, the combination and division of cells will happen anyway. This will cause the embedded signal to be distorted, and the probability of recognizing the embedded signal will be reduced. To deal with the distortion of the signal, we present a recovery mechanism that robustly recognizes the embedded signal.

The combination or division of the cells for different units of the signal can be categorized into four types. Fig. 7(a) illustrates two types of the cell division for the unit of the signal, and Fig. 7(b) illustrates the two types of the cell combination for different units of the signal. Let $C = \{C_0, C_1, \ldots, C_i, \ldots, C_{m-1}\}$ be the cell numbers recorded in the circuit queue at the entry onion router, and C_i ($i \in [0, m-1]$) is the number of the cells, which is a positive integer. Recall the original signal is denoted as $S = \{S_0, S_1, \ldots, S_j, \ldots, S_{n-1}\}$. Let S_j be the jth signal bit, S'_j as the part of the jth signal bit, and let S_x be the integral signal bits or a remaining signal bit in the packet or a null signal bit. Type-I distortion indicates that the original signal S_j is divided into $k+1$ separate cells. Fig. 8 illustrates an example for Type I with $k = 1$. Suppose signal S_j is bit "1"; the number of cells should be 3. As a matter of fact, the attacker at the entry onion router records that C_i is 1 and C_{i+1} is 2, i.e., three cells for signal S_j are divided into one cell and two cells. Moreover, signal S_j may also be divided into three separate cells, i.e., $k = 2$. Type-II distortion indicates that the last part of S_j is merged with the following signal(s) S_x. Fig. 8 illustrates an example for Type III with $k = 1$. Suppose signal S_j is bit "1" and S_x is a integral signal S_{j+1} for "0" bit. However, the attacker records that C_i is 1 and C_{i+1} is 3, i.e., the part of S_j is merged with the followed signal S_{j+1}. Type-III distortion indicates that k original signals are merged into a signal packet. Fig. 8 illustrates an example for Type III with $k = 2$. If S_j, S_{j+1} and S_{j+2} are "010," the attacker records that C_i is 5. In this case, the cells belonging to three signal units are merged

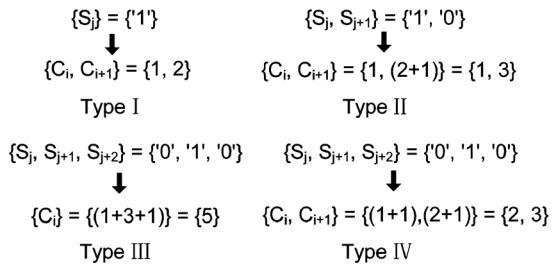


Fig. 8. Examples of signal division and combination.

all together. Type-IV distortion indicates that a part of S_{j+k} is merged into the following cells. Fig. 8 illustrates an example for Type III with $k = 2$. If signal S_j, S_{j+1} and S_{j+2} are "010" bits, C_i and C_{i+1} will be recorded as 2 and 3, respectively. We give simple examples of four types of division and combination listed above. The division or combination of the cells in these types may be even more complicated on Tor.

Signal Detection Schemes: To deal with those types of combination and separation, we propose our detection scheme. Algorithm 1 in Appendix A shows the recovery mechanism. If the number of cells recorded in the circuit queue is smaller than the number of cells of the original signal, the signals are recovered as either Type I or Type II. Suppose the number of cells recorded in the circuit queue is larger than the number of cells for carrying the signal; these recovered signals will be either Type III or Type IV depending on the condition whether there is S_x in C_{i+1}. When the signals are recovered in these Types with $k \leq 2$, we consider that these signals are successfully identified. Otherwise, the signals cannot be identified.

IV. Extension and Discussion

In this section, we study various issues, including the impact of controlling both entry and exit onion routers, how an attacker uses only Tor exit routers for launching the attack, and the detectability and other impacts of the attack.

A. Impact of Controlling Both Entry and Exit Onion Routers

We now investigate the impact of controlling both entry and exit onion routers. We assume that the attacker needs to set up malicious onion routers in the Tor network. As mentioned in [23], there are four types of onion routers at the Tor network—namely, entry router, middle router, exit router, and both entry and exit router (denoted as EE router). In the cell-counting-based attack, the attacker controls a number of onion routers as either entry routers or exit routers. In order to understand the impact, we need to evaluate the probability that a TCP stream traverses both the malicious entry onion router and exit onion router, given that a number of routers in Tor are malicious and controlled by attackers.

To ensure the performance of circuits, Tor adopts weighted bandwidth routing algorithms. First, the client chooses an appropriate exit onion router *OR3* from the set of exit routers, including the pure exit routers and EE routers. The bandwidth of exit routers is weighted as follows. Assume that the total bandwidth is B, the total exit bandwidth is B_{E}, and the total entry bandwidth is B_{G}. If $B_{\mathrm{E}} < (B/3)$, i.e., the bandwidth of exit routers is scarce, the exit routers will not be considered for nonexit use. The bandwidth of EE routers are weighted by $W_{\mathrm{G}} = 1 - (B/3B_{\mathrm{G}})$, where W_{G} is the bandwidth weight of entry routers and $B_{\mathrm{G}} > (B/3)$. If $B_{\mathrm{G}} < (B/3)$, then $W_{\mathrm{G}} = 0$. The probability of selecting the ith exit router from the exit set is $Bi_{\mathrm{E}}/(B_{\mathrm{E}} + B_{\mathrm{EE}} \cdot W_{\mathrm{G}})$, where B_{EE} is the total bandwidth of EE routers. Second, the client chooses an appropriate entry onion router *OR1* from the set of entry routers, including the pure entry routers and EE routers. To ensure sufficient entry bandwidth, if $B_{\mathrm{G}} < (B/3)$, the entry routers will not be considered for nonentry use. Then, the probability of selecting the ith entry router from the entry set is $Bi_{\mathrm{G}}/(B_{\mathrm{E}} + B_{\mathrm{EE}} \cdot W_{\mathrm{E}})$, where $W_{\mathrm{E}} = 1 - (B/3B_{\mathrm{E}})$ is the exit bandwidth weight and Bi_{G} is the ith bandwidth in the entry set. If $B_{\mathrm{E}} < (B/3)$, then $W_{\mathrm{E}} = 0$. Eventually, the client chooses the middle from the rest of Tor routers.

Assume that we configure EC2 nodes as malicious entry, exit, or EE routers. Denote the number of malicious exit routers as e_1, the number of malicious entry routers as e_2, and the number of the malicious EE routers as e_3, where $e_3 = k - e_1 - e_2$. Based on the above weighted bandwidth selection algorithm, the weight can be derived by

$$W_{\mathrm{E}} = \begin{cases} 1 - \frac{B}{3\cdot(B_{\mathrm{E}}+B_{\mathrm{EE}}+(e_1+e_3)*b)} : & W_E > 0 \\ 0 : & W_{\mathrm{E}} \leqslant 0 \end{cases} \quad (6)$$

$$W_{\mathrm{G}} = \begin{cases} 1 - \frac{B}{3\cdot(B_{\mathrm{G}}+B_{\mathrm{EE}}+(e_2+e_3)*b)} : & W_{\mathrm{G}} > 0 \\ 0 : & W_{\mathrm{G}} \leqslant 0. \end{cases} \quad (7)$$

Then, the catch probability can be calculated as follows:

$$\begin{aligned} P(e) = & \frac{e_1 \cdot b}{B_{\mathrm{E}} + W_{\mathrm{G}} * (B_{\mathrm{EE}} + e_3 \cdot b) + e_1 \cdot b} \\ & \cdot \frac{(W_{\mathrm{E}} \cdot e_3 + e_2) \cdot b}{B_{\mathrm{G}} + W_{\mathrm{E}} * (B_{\mathrm{EE}} + e_3 \cdot b) + e_2 \cdot b} \\ & + \frac{W_{\mathrm{G}} \cdot e_3 \cdot b}{B_{\mathrm{E}} + W_{\mathrm{G}} * (B_{\mathrm{EE}} + e_3 \cdot b) + e_1 \cdot b} \\ & \cdot \frac{(W_{\mathrm{E}} \cdot (e_3 - 1) + e_2) \cdot b}{B_{\mathrm{G}} + W_{\mathrm{E}} * (B_{\mathrm{EE}} + (e_3 - 1) \cdot b) + e_2 \cdot b}. \end{aligned} \quad (8)$$

According to the above formula, we could derive the maximum P and the corresponding number of exit routers and entry routers [24].

In our recent study [24] shown in Fig. 9, we showed that by injecting around 4% of onion routers with long uptime and high bandwidth, the attack can confirm over 60% of the communication sessions over Tor.[1] We consider two strategies. In Scheme 1, we donate nodes such as those from Amazon EC2 as either Tor exit routers or entry routers (not as EE routers). In Scheme 2, we configure EC2 nodes as entry, exit, or EE sentinels. We can see that these two schemes achieve similar results. Note that since TorFlow [25] can measure the real bandwidth of the Tor nodes, the attacker should rent sufficient bandwidth for each EC2 node instead of making fake bandwidth advertisement. Because of the pay-as-you-go model of the cloud computing, such a bandwidth rent is feasible to malicious organizations or people with modest power.

According to previous research [26], Tor will suffer severe TCP performance degradation if it adopts the random path selection strategy to reduce the impact of the attack. The bandwidth of 90% of Tor routers is less than 350 kB/s [24]. Suppose that a client uses random path selection strategy, the probability

[1]Note the fact is true for any powerful traffic confirmation attack as well as the proposed attack.

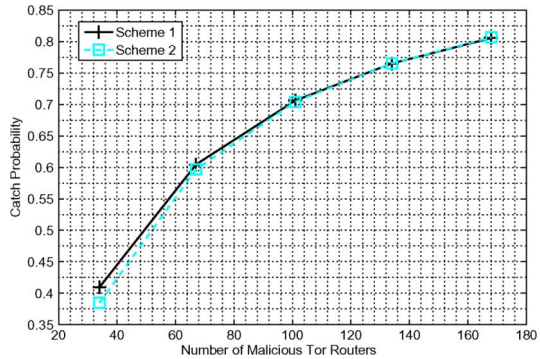


Fig. 9. Probability that a circuit chooses the malicious routers as entry and exit routers versus number of malicious Tor routers [19].

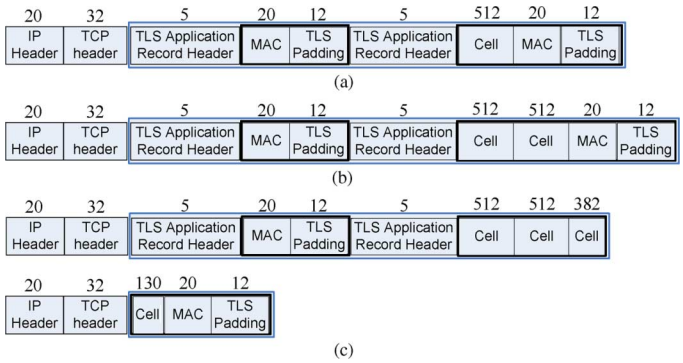


Fig. 10. Packet format. (a) Packet format of 1 cell. (b) Packet format of 2 cell. (c) Packet format of 3 cell.

Fig. 11. TLS header.

that it selects Tor routers with low bandwidth for the circuits is around 90%. Obviously, the low-bandwidth Tor router will be the bottleneck of the circuit.

B. Controlling Exit Onion Routers Only

If the attacker does not control entry onion routers, the cell-counting-based attack can still be successful. An attacker can sniff the packets transmitted between an entry onion router and a client. The attacker may recover the embedded signal based on the size of the packet. In this way, the number of required malicious routers in Tor can also be reduced while the attack still has a desired impact.

We now introduce the structure of the IP packet that envelops the cell(s) and passes along the network. Without loss of generality, we assume that MTU is 1500 B. Fig. 10(a) illustrates the structure of IP packet that envelops one cell, including an IP header, a TCP header, an empty TLS application record, and a TLS application record of enveloping one cell. The TLS record packet incorporates a TLS header (5 B), a TLS message (not to exceed 2^{14} B), a MAC (Message Authentication Code, 20 B), and a TLS padding (12 B). Fig. 11 illustrates the header of the TLS packet, with the length of 5 B. The field of content type identifies the record-layer protocol type contained in this record, with the length of 1 B. In our case, we are concerned with the

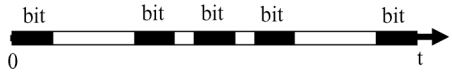


Fig. 12. Time-hopping technique.

TLS application record, with content type of 23. The field of version identifies the major or minor version of TLS for the contained message, with the length of 2 B. The field of length identifies the length of protocol message(s), not to exceed 2^{14} B.

Fig. 10(b) illustrates the structure of IP packet that envelops two cells and has a length of 1150 B. Because an IP packet that envelops three cells exceeds the MTU (1500 B), this IP packet will be segmented; one segment has the packet of 1500 B, and the other segment has the packet of 214 B. Fig. 10(c) illustrates the structure of IP packet that envelops three cells and is segmented. Hence, the attacker can map "0" bit of the signal to one IP packet, with the length of 638 B. By appropriately choosing a delay interval at the exit onion router, the "1" bit of the signal will have two cases: two IP packets with one cell [shown in Fig. 10(a)] and two cells [shown in Fig. 10(b)], i.e., the signal is divided as Type I with $k = 1$, as well as two IP packets with three cells [shown in Fig. 10(c)], which is neither divided nor combined. Therefore, from packet size pattern, the attacker is still able to recognize the signal embedded in the IP packet stream by using our signal detection mechanism. Actually, the fact that multiple cells can be packed into a packet guarantees the correct signal encoding via the variation of the cell count. When such a packet arrives at the TLS buffer, those cells form a group, which is read into the circuit queue. This is our mechanism that generates a signal bit "1" or "0."

C. Attack Detectability

The proposed cell-counting-based attack is difficult to detect. As we know, the attack transmits a short and secret random signal known only to the attackers. It is difficult to detect within the target traffic. Based on the evaluation data shown in Section VI, the success of this attack requires only a short secret signal—such as 5 b—while achieving a detection rate of almost 100% and a false positive rate of $1/10^5$. It would be hard to classify such a short sequence of random signals as the attack sequence in bursty network traffic.

To further improve the attack invisibility, we adopt the time-hopping-based signal embedding technique, which can greatly reduce the probability of interception and recognition [27]. Fig. 12 illustrates the principle of the time-hopping technique. For the time hopping, there exist random intervals between signal bits. At the exit onion router, the duration of those intervals are varied according to a pseudorandom control code, which is known to only the attackers. To recover the signal at the entry onion router, an accomplice of the attacker can use the same secret control code to help position the signal bits and recover the whole signal. Intuitively, if the interval between the bits is large enough, the inserted signal bits appear sparse within the target traffic, and it is difficult to determine whether groups of cells are caused by network dynamics or by intention. Therefore, the secret signal embedded into the target traffic is no different than the noise. In addition, when a malicious entry node has confirmed the communication relationship, it can separate the group of cells by adding delay between the cells so that not even the client can observe the

embedded signal. In Section VI, we demonstrate the effectiveness of this time-hopping-based technique, and the detailed approach is shown in Algorithm 2 in Appendix A.

In our proposed attack, a secret signal is embedded into the target traffic, which implies a secret sequence of groups of one and three cells. One may be concerned that if the sequence of groups of one and three cells is unnatural and the entry node is honest and aware of the attack, it will detect the sequence and thus distinguish the traffic flow *with* an embedded signal from a flow *without* a signal. However, with the time-hopping technique, groups of one and three cells are separated by random intervals, and it is hard to differentiate them from those caused by network dynamics. As a side note, the false positives in detecting signal bits in Section VI's figures imply that normal network traffic does have groups of one and three cells caused by network dynamics. In addition, since the embedded signal is very short and only known to attackers, we conjecture that it is very difficult to distinguish traffic with embedded signals from normal traffic based on this very short secret sequence of cell groups.

D. Difference From Existing Attacks

The proposed cell-counting-based attack may dramatically degrade anonymity that Tor maintains. Different from other existing attacks, the cell-counting-based attack is accurate, efficient, and difficult to detect. This attack requires much fewer packets and incurs little overhead while achieving a higher detection rate than most traffic analysis attacks, including traffic confirmation attacks in [10], [13], [17], [28], and [29]. Since this attack utilizes the atomic unit of a traffic flow, i.e., cells/packets (and their size), this attack is highly efficient and can confirm very short communication sessions with only tens of cells. Although the tagging-based attack [19], [30] may require few packets, it tears down the Tor circuits and is relatively easy to detect. A simple passive cell-counting attack may count the cells at points of exit and entry onion routers and correlate the counting. However, there is no guarantee of detection rate and false positive rate because of the large number of connections running through Tor. In addition, our attack achieves a low false positive rate with a very small amount of target traffic as demonstrated in Section VI. Therefore, as a powerful traffic confirmation attack, the proposed attack poses a great challenge against Tor.

E. Countermeasures

We now discuss possible countermeasures. It is also difficult for Tor to defeat the cell-counting attack. One possible countermeasure is that Tor routers add delay between cells in order to disrupt malicious cell groups. However, choosing such a delay will be very challenging. A too short delay cannot separate cells (at the network layer), while a long delay may dramatically degrade Tor's performance, which is already the biggest bottleneck of using Tor [22], [26], [31]. A second way to reduce the impact of the proposed attack is to use purely random routing algorithms and reduce the chance of traffic flows passing malicious Tor onion routers. However, such a random routing algorithm will also degrade Tor performance. Its effect is also very limited since the attacker can inject more malicious routers into Tor to increase the impact.

Dummy traffic may be used to distort the timing of the signal. A constant rate padding along a circuit may incur too much overhead. Levine *et al.* [8] investigated a defensive dropping scheme, in which dummy traffic can be randomly dropped at the intermediated routers. An end-to-end defensive dropping cannot be applied to Tor directly. Tor adopts Advanced Encryption Standard Counter Mode (AES-CTR) to encrypt the cells. The AES counter at each onion router and onion proxy is synchronized. Defensive dropping will disrupt this AES counter and cause decryption errors at the onion proxy or the exit routers [30]. These errors will tear down the circuits. Shmatikov *et al.* [15] proposed an adaptive padding scheme by injecting dummy packets into statistically unlikely gaps in the packet flow, destroying timing fingerprints without adding any latency to the application traffic. However, in our case, the attacker controls the exit router, and the signal can be embedded in the dummy traffic as well.

This paper provides guidance to anonymous protocol design and implementation. To design an anonymous communication system, we have to consider the impact of the design on all protocol layers. For example, Tor implements an overlay protocol and preserves equal-sized cells on the application layer. However, the equal-sized cells on the application layer cannot guarantee that packets on the network layer are also equal-sized. Hence, the equal-sized cells on the application layer cannot guarantee the anonymity provided by Tor. Indeed, our attack exploits the Tor protocol's impact on the network layer.

V. Analysis

In this section, we show the analytical results for the accuracy and efficiency of the cell-counting-based attack. For attack accuracy, we derive closed formulas for detection rate and false positive rate. Our theoretical analysis shows that the detection rate is a monotonously increasing function with respect to the delay interval and is a monotonously decreasing function of the variance of one way transmission delay along a circuit. Our experimental results in Section VI match the theoretical results well.

A. Detection Rate

We view that the major factor causing detection error is network dynamics, which leads to combination and division of cell groups. Our analysis is based on the network configuration described in the second paragraph of Section II-B. The round-trip delay between two onion routers can be modeled by a log-normal distribution [32]. We first investigate the properties of the log-normal distribution and then use the delay model to derive detection rate analytically.

A log-normal random variable has the property that its logarithm has a *Gaussian* distribution. Let X_i be a Gaussian random variable with the probability density function (PDF), we have

$$f_{x_i}(x) = \frac{1}{\sigma_{x_i}\sqrt{2\pi}} e^{\frac{-(x-\mu_{x_i})^2}{2\sigma_{x_i}^2}} \tag{9}$$

where μ_{x_i} and σ_{x_i} are mean and standard derivation, respectively. Let $X_i = \ln L_i$, where L_i is a random variable with log-normal distribution and the PDF of L_i is given by

$$f_{L_i}(l) = \frac{1}{l\sigma_{x_i}\sqrt{2\pi}} e^{\frac{-(ln(l)-\mu_{x_i})^2}{2\sigma_{x_i}^2}}. \tag{10}$$

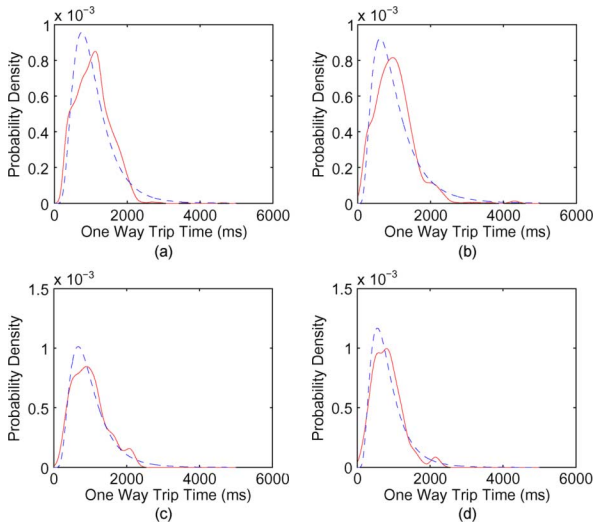


Fig. 13. One-way trip time probability density function. (a) Germany. (b)–(d) US.

Let L_1 be the log-normal random variable of the delay between *OR3* and *OR2*, and L_2 be the log-normal random variable of the delay between *OR2* and *OR1*. Following the widely used assumption that a sum of independent log-normal random variables is well approximated by another log-normal random variable, we have

$$L = L_1 + L_2 = e^{X_1} + e^{X_2} \approx e^H \tag{11}$$

where the random variable H possesses a Gaussian distribution. Therefore, the round-trip delay between *OR3* and *OR1* is also a log-normal distribution L. Since L follows a log-normal distribution, the arrival time of the signal at the entry onion router is approximately $L/2$, which is a log-normal distribution as well. This fact is formally proved in Appendix B.

We have experimentally measured one-way trip time along the circuit and verified this fact. In our experiments, the client sends a cell to the server every 10 s via the *OP*. We change the configuration of the client to select our entry node and exit node for its circuits. We use Network Time Protocol (NTP) to synchronize entry node and exit node.[2] The entry node and exit node record the timestamp of the incoming cells. The middle nodes are selected randomly by the client. Therefore, the difference of the timestamps recorded in entry nodes and exit nodes are one-way trip time between entry nodes and exit nodes. Fig. 13 shows that the realistic data can be approximated by the log-normal distribution. Note that in this figure, the solid line is the PDF derived from the realistic data. The dashed line is the estimated log-normal PDF by using maximum likelihood estimation (MLE). The middle node in the experiments producing Fig. 13(a) is in Germany, while all the other middle nodes for Fig. 13(b)–(d) are in the US. From these figures, we can see that the empirical curves match the estimated log-normal distribution curves well.

[2]NTP ver. 4 can usually maintain a time accuracy of 10 ms over the public Internet and can achieve an accuracy of 0.2 ms or better in local area networks. We obtain statistics to show the trend of one-way delay between an entry node and exit node, and the accuracy provided by NTPv4 is sufficient for our experiments.

Now we derive the detection error rate. Let n be the length of the original signal, and the arrival time of the signals at the entry onion router be $T = \{T_{s_0}, T_{s_1}, \ldots, T_{s_{n-1}}\}$. Let the delay interval between the two bits of the signal be Δt. Because cells associated with the neighboring signal bits can be combined (when $T_{s_i} > T_{s_{i+1}} + \Delta t$), the probability of error becomes

$$P_{\mathrm{e}} = \Pr\left(T_{s_i} > T_{s_{i+1}} + \Delta t\right) = \Pr\left(T_{s_i} - T_{s_{i+1}} > \Delta t\right). \tag{12}$$

Letting $Z = T_{s_i} - T_{s_{i+1}}$, we have

$$P_{\mathrm{e}} = \Pr(Z > \Delta t) = 1 - \Pr(Z \le \Delta t). \tag{13}$$

Detection rate P_{D} is defined as the probability that a 1-b original signal is recognized correctly. We have

$$P_{\mathrm{D}} = 1 - P_{\mathrm{e}} = \Pr(Z \le \Delta t). \tag{14}$$

Let $T_{s_i} \doteq X$ and $T_{s_{i+1}} \doteq Y$. We have $Z = X - Y$. Assume T_{s_i} and $T_{s_{i+1}}$ are independent and identically distributed (i.i.d.). X and Y are i.i.d. as well. Let μ and σ be mean and standard deviation of the variable ($\ln X$ or $\ln Y$). Because

$$F_Z(z) = \Pr(Z \le z) = \Pr(X - Y \le z) \tag{15}$$

$$= \int_0^{+\infty} \left(\int_0^{z+y} f(x,y)dx \right) dy \tag{16}$$

then

$$\Pr(Z \le \Delta t)$$

$$= F_Z(\Delta t) \tag{17}$$

$$= \int_0^{+\infty} \left(\int_0^{\Delta t+y} f(x,y)dx \right) dy \tag{18}$$

$$= \int_0^{+\infty} f(y) \left(\int_0^{\Delta t+y} f(x)dx \right) dy \tag{19}$$

$$= \int_0^{+\infty} f(y) \left(\frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{\ln(\Delta t + y) - \mu}{\sigma\sqrt{2}} \right) \right) dy \tag{20}$$

where $\mathrm{erf}(x) = (2/\sqrt{\pi}) \int_0^x e^{-t^2} dt$.

In addition, the first derivative of function $F_Z(\Delta t)$ is given by

$$F'_Z(\Delta t) = \frac{1}{2} \int_0^{+\infty} f(y) \frac{1}{(\Delta t + y)\sigma\sqrt{2}} \frac{2}{\sqrt{\pi}} e^{-\left(\frac{\ln(\Delta t+y)-\mu}{\sigma\sqrt{2}}\right)^2} dy \tag{21}$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_0^{+\infty} \frac{1}{\Delta t + y} e^{-\left(\frac{\ln(\Delta t+y)-\mu}{\sigma\sqrt{2}}\right)^2} f(y)dy. \tag{22}$$

Since $\Delta t > 0$ and $y > 0$, we have $F'_Z(\Delta t) > 0$. Hence, $P_{\mathrm{D}} > 0$ and P_{D} is a monotonously increasing function in terms of Δt. Therefore, the larger the delay interval we choose, the higher the detection rate that will be achieved. This result is also validated by our real-world experimental data in Section VI.

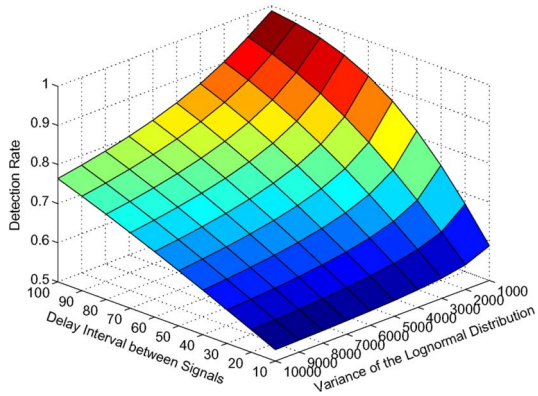


Fig. 14. Theoretical detection rate versus delay interval and variance of the log-normal distribution.

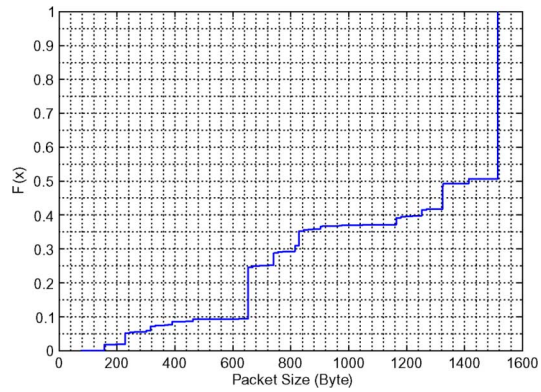


Fig. 15. Empirical cumulative distribution function (CDF) of packet size.

Detection rate $P_{\mathrm{D},n}$ is defined as the detection rate for an n-bit original signal. Given P_{D} for detection rate for 1-b original signal, we have

$$P_{\mathrm{D},n} = (P_{\mathrm{D}})^n \tag{23}$$

which is a monotonously increasing function with the delay interval as well.

Fig. 14 illustrates the theoretical results based on the above theoretical analysis, i.e., (17). It shows the relationship among the theoretical detection rate, delay interval (ms), and variance of the log-normal distribution. Assume that the mean of the log-normal distribution μ is 700 ms. We have two observations from Fig. 14: 1) the theoretical detection rate is a monotonously increasing function with respect to the delay interval Δt; 2) the theoretical detection rate is a monotonously decreasing function with respect to the variance σ^2 of the log-normal distribution. Our experimental results in Section VI match these observations well and validate our theoretical analysis.

B. False Positive Rate

When there is no signal embedded into the target traffic, there is the possibility that the detection could reach an incorrect decision. Packets in the normal traffic would have different sizes. Let the probability of one cell packed in an IP packet be p_0 (which will be recognized as signal bit "0"). Let the probability of three cells packed in the packet be p_1 (which will be recognized as signal bit "1"). Let p_2 be the probability that packets have other sizes. We have $p_2 = 1 - p_0 - p_1$.

The false positive rate $P_{\mathrm{F},n}$ for recognizing an n-bit signal can be calculated by

$$P_{\mathrm{F},n} = \left(\frac{p_0 + p_1}{2}\right)^n = \left(\frac{1 - p_2}{2}\right)^n. \tag{24}$$

To obtain the empirical distribution of IP packet size for the traffic within the Tor network, we downloaded a file with the size of 20 M using the Tor network. Fig. 15 shows the cumulative probability function for the packet size in normal traffic. It shows that the sum of p_0 and p_1 is around 0.5. Then, we have

$$P_{\mathrm{F},n} \approx \left(\frac{0.5}{2}\right)^n = \left(\frac{1}{4}\right)^n. \tag{25}$$

Therefore, we will have a lower false positive rate, as the original signal length n becomes longer. Given the false positive rate $P_{\mathrm{F},n}$ in the above formula, we can determine the original signal length n. For example, given the false positive rate of 1.5% (or 0.4%), we can use an original signal of length 3 (or 4). In our extensive experiments in Section VI, we observed even much lower false positive rate.

C. Attack Capacity

We can use the information-theoretical model to analyze the efficiency of the cell-counting-based attack. Recall that in this attack, the attacker at the exit onion router embeds a bit of signal, and the attacker at the entry router recognizes a correct bit or a wrong bit information. When bit information is mistakenly recognized, we call the signal bit to be "erased." Hence, the cell-counting-based attack technique uses the host traffic from the exit onion router to the entry router as a covert channel to transmit an invisible signal. Using the concept of channel capacity, we can obtain efficiency of our investigated attack. Channel capacity defined by Shannon gives a theoretical bound for measuring the information transmission capability over a noisy channel [33].

Assume the channel model in our system is a discrete and memoryless channel (DMC). This attack can be modeled as a binary erasure communication channel as shown in Fig. 16, where X presents the transmission signal, Y presents the received signal, and p represents the probability of transmitting bit 1. Recall that the network congestion or delay can result in the erasure signal (i.e., either 1 or 0) in our case. Let the probability that one bit of signal is "erased" be P_{e} and t_i be the amount of time to transmit one input bit x_i across the DMC channel. The random variable $\mathcal{T}$ is the time to send such a bit. Note that $\mathcal{T}$ also includes the network delay caused by network dynamics. The mean of $\mathcal{T}$ is represented by $E(\mathcal{T})$. The mutual information in units of *bits per second* $I_t(X, Y)$ considering the transmission time cost for a channel is

$$I_t(X, Y) = \frac{I(X, Y)}{E(\mathcal{T})}. \tag{26}$$

Then, the capacity C_t in units of bits per second for a DMC [34] is given by

$$C_t = \max_p \frac{I(X, Y)}{E(\mathcal{T})}. \tag{27}$$

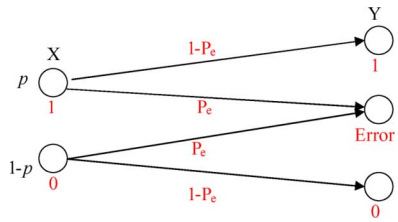


Fig. 16. Channel model.

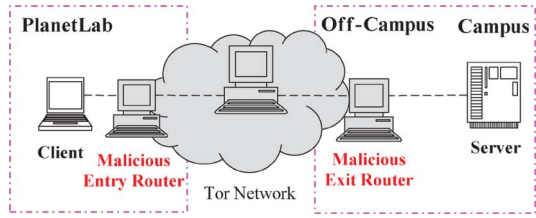


Fig. 17. Experiment setup.

Based on the channel model shown in Fig. 16, we know that $\max I(X,Y)$ can be derived by

$$\begin{aligned}
&\max_p I(X,Y)\\
&\quad= \max_p H(X) - H(X|Y) &(28)\\
&\quad= \max_p H(X) - \sum p(y)H(X|Y=y) &(29)\\
&\quad= \max_p H(X)\\
&\qquad - [p(1-P_e)H(X=|Y=0)\\
&\qquad\quad + (1-p)(1-P_e)H(X|Y=1)\\
&\qquad\quad + (pP_e + (1-p)P_e)\,H(X|Y=\text{error})] &(30)\\
&\quad= \max_p H(X) - [p(1-P_e)*0 + (1-p)(1-P_e)\\
&\qquad\qquad * 0 + P_e H(X)]\\
&\quad= \max_p H(X)(1-P_e)\\
&\quad= 1 - P_e &(31)\\
&\quad= P_{\mathrm{PD}} &(32)
\end{aligned}$$

and we have

$$C_t = \frac{P_{\mathrm{D}}}{E(\mathcal{T})}. \tag{33}$$

Note that $E(\mathcal{T})$ determines how quickly the attacker can transmit one bit of signal. In our case, it is determined by Δt along with other factors. Recall that Δt is the delay interval between the two bits of the transmitted signal. From the above analysis, we also know that P_{D} is a monotone increasing function of Δt. The above formula provides a few important insights into the capability of cell-counting-based attack. With the increase of Δt, a larger P_{D} can improve the capacity, but it causes a larger $E(\mathcal{T})$ and deteriorates the capacity. With the decrease of Δt, it can obtain a smaller $E(\mathcal{T})$ to improve the capacity, but it causes a smaller P_{D} and deteriorates the capacity. From Fig. 14, we observe that the decline of P_{D} is slower than the decline of Δt. Consequently, with the increase of Δt, the capacity will increase. Nevertheless, the capacity will decrease when Δt reaches a certain level.

VI. Experimental Evaluation

We have implemented the cell-counting-based attack presented in Section III against Tor [35]. In this section, we use real-world experiments to demonstrate the feasibility and effectiveness of this attack. All the experiments were conducted in a controlled manner, and we experimented on TCP flows generated by ourselves in order to avoid legal issues.

A. Experiment Setup

In our experiment setting illustrated in Fig. 17, we deployed two malicious onion routers as the Tor entry onion router and exit onion router. The entry onion router and client (Alice) located in Asia are deployed on PlanetLab [36]. The server (Bob) is located at one university campus in North America, and the exit onion router is at an off-campus location in North America as well. All computers are on different IP address segments and connected to different Internet service providers (ISPs). Fig. 17 shows the experiment setup.

We modified the Tor client code for attack verification purpose. The Tor client will intend to setup circuits through the designated malicious exit onion router and entry onion router shown in Fig. 17. The middle onion router is selected using the default routing selection algorithm released by Tor. As we stated earlier, the cell-counting-based attack intends to confirm whether the client (Alice) communicates with the server (Bob). For verification purpose, we set up a server (Bob) and download a file from the client (Alice). The downloading software at the client is the command line utility *wget*. By configuring *wget*'s parameters of *http_proxy* and *ftp_proxy*, we let *wget* download files through *Privoxy*, the proxy server used by Tor. By using the Tor configuration file and manipulatable parameters, such as *EntryNodes, ExitNodes, StrictEntryNodes,* and *StrictExitNodes* [23], we let the client choose both the malicious entry and exit onion routers along the circuit.

B. Experimental Results

To obtain the empirical property of IP packet size for the traffic within the Tor network, we downloaded a file with the size of 20 M using the Tor network. Fig. 15 shows the empirical cumulative probability function (CDF) of the IP packet size in the traffic. As shown in Fig. 5, we know that the packets with non-MTU size are around 50%. This validates that the size of packets transmitted over the Tor is dynamic. Consequently, it also indicates that our embedded signal will be hidden in the normal traffic and hard to be detected by victims.

To validate the accuracy of the cell-counting-based attack, we let the client download 30 files in our experiments. The size of each file is around 10 MB. At the exit onion router, we generate a random signal with 100 b. When the target traffic from server Bob arrives at the exit onion router, we vary the number of cells in the circuit and embed the signal into the variation of the cell count during a short period in the target traffic. At the entry onion router, the cells in the circuit queue are recorded in the log, and the recovery mechanisms will be applied to recognize the embedded signal. In addition, we chose different thresholds and types in our recovery mechanism as discussed in Section III-C. In particular, we chose to recover Type I and III with $k = 1$

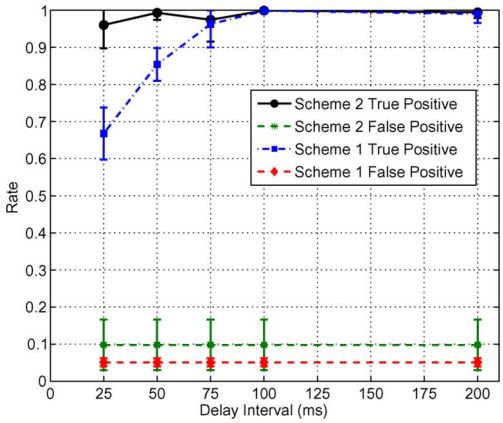


Fig. 18. Detection rate versus delay interval (Note: The rate is for detecting one bit).

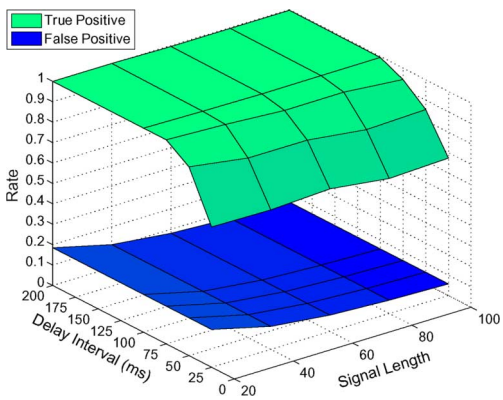


Fig. 19. Detection rate versus delay interval and signal length with detection scheme 1 (Note: The rate is for detecting one bit).

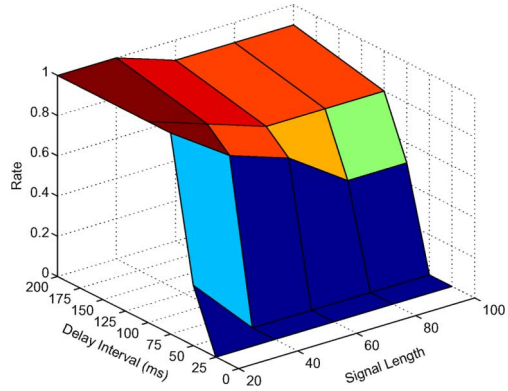


Fig. 20. Detection rate versus delay interval and signal length with detection scheme 1.

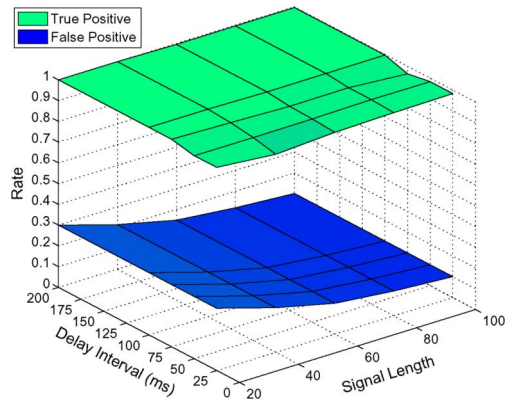


Fig. 21. Detection rate versus delay interval and signal length with detection scheme 2 (Note: The rate is for detecting one bit).

as detection scheme 1. Moreover, we chose to recover all types with $k = 2$ as detection scheme 2.

When we evaluate the false positive rate, the client downloads 30 files via Tor again. However, no signal is embedded into the traffic at the exit onion router. Denote the traffic with no signal as clean traffic. We generate a 100-b random signal and apply detection schemes 1 and 2 to the clean traffic collected at the entry onion router. By checking how many bits of this signal show up in the clean traffic, we can calculate the false positive rate.

We conduct the above experiment to evaluate the true positive and false positive by using a 100-b random signal. Fig. 18 illustrates the correlation between the detection rate (true positive) and the delay interval for transmitting cells associated to different units of the signal. As we can see from this figure, the detection rate will increase dramatically when the delay interval is slightly increased in two detection schemes. As expected, the detection rate of scheme 2 is higher than scheme 1 with a slightly increasing false positive rate, while the overall false positive rate for each scheme is a fixed value. When the delay interval approaches 100 ms, the detection rate of two schemes approaches 100%. All these findings validate that our investigated attack can significantly degrade the anonymity service provided by Tor.

Fig. 19 illustrates the detection rate in terms of signal length and the delay interval for scheme 1. Note that the detection rate in Fig. 19 is for detecting one bit. As we can see from this figure, when we increase the signal length from 20 to 100, the detection rate will be slightly decreased, and the false positive rate will be constantly very low (less than 5%). When the signal length is 20, and the delay interval between signals is 100 ms, 100% detection rate can be achieved. In addition, Fig. 20 illustrates the detection rate for detecting the whole signal in terms of signal length and the delay interval for scheme 1. When the signal length is 20, and the delay interval between the signals is 100 ms, a detection rate of 100% can be achieved. In Fig. 20, the false positive approaches 0%, and this matches our theoretical analysis in Section V-B. This validates that the investigated attack only requires tens of cells and is highly efficient to confirm very short communication sessions on Tor. Fig. 21 illustrates the detection rate in terms of signal length and the delay interval for scheme 2. Note that the rate shown in Fig. 21 is for detecting one bit. The false positive decreases quickly with the increasing signal length. Additionally, the detection rate can approach 100% with the delay interval of 100 ms and signal length of 100 with a low false positive. Fig. 22 illustrates the detection rate for detecting the whole signal in terms of signal length and the delay interval for scheme 2. The detection rate can approach 100% with the delay interval 100 ms and signal length 20, and the false positive approaches 0%.

To further improve the detectability of cell-counting-based attack, we also investigated the improved encoding mechanism, called the hopping-based encoding, which randomly

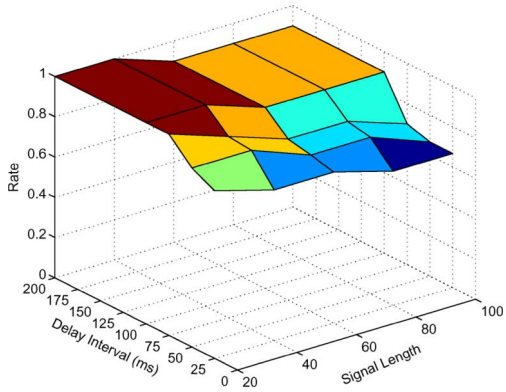


Fig. 22. Detection rate versus delay interval and signal length with detection scheme 2.

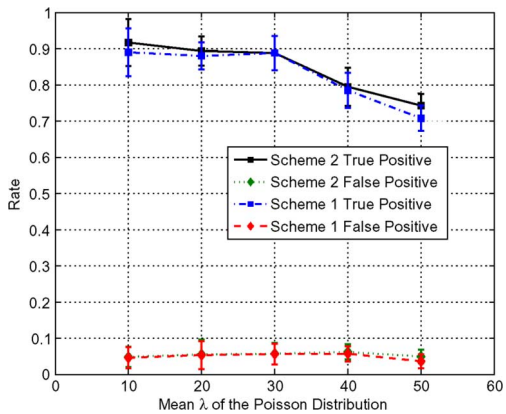


Fig. 23. Correlation between detection rate and mean λ of the Poisson distribution (Note: The rate is for detecting one bit).

embeds units of a signal into the target traffic, as introduced in Section IV-B. For this encoding scheme, we generate an array $Q[1 \times n]$ by using a Poisson distribution with a mean λ. We first send the number of cells $Q[i]$ without embedding signals, and then embed a signal bit. In this set of experiments, we also chose a signal length of 100. Since the units of the signal are embedded randomly in a hopping fashion in the time domain, it is hard for the multiflow attack [37] to detect the embedded signal in the traffic. Fig. 23 illustrates the relationship between detection rate (true positive) and the mean λ of nonwatermarked cells (which corresponds to the random time interval. No signal is embedded into those cells). From Fig. 23, we can see that this improved encoding scheme can still achieve very high detection rates along with a very low false positive rate. Since this new encoding scheme does not embed the signal into all *CELL_RELAY_DATA* cells, the attack will require more cells in order to be successful. Additionally, based on Algorithm 1, we use Algorithm 2 to recognize the signal embedded in the sparsely encoded cells.

We also use *tcpdump* to capture the IP packets transmitted between the entry node and the client and demonstrate that an attacker may also use packet size to recognize the embedded signal. Fig. 24 illustrates the variance of IP packet size. As we can see, there are three types of IP packet sizes, and the corresponding packet structures are shown in Fig. 10. According to detection scheme 1 (blind detection approach), we can map bit "0" of the signal to the IP packet size of 638 B. Bit "1"

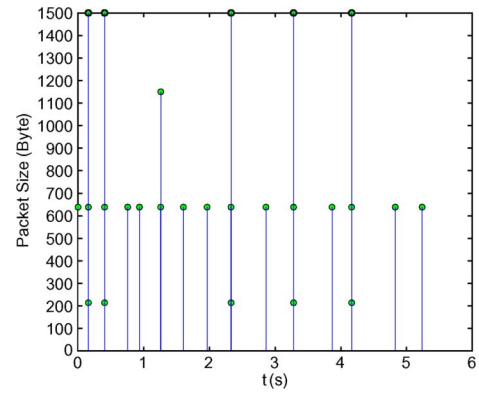


Fig. 24. Variance of IP packet size.

of the signal has two cases: IP packet of 638 B followed by IP packet of 1150 B, as well as one IP packet of 1500 B and of 214 B, as we discussed in Section IV-B. Therefore, we can decode the signal between 0 and 1 s as "0010100." Note that since the delay interval is very small among the second (638 B), the third (1500 B), and the fourth (214 B) IP packets, they are mostly overlapped in the figure. As we know, the packet drops will incur TCP retransmissions, and it may result in a distorted signal. One way to address the packet drop issue is to use tcpdump and check the TCP sequence numbers for assisting the signal recovery. A second way is to increase the delay interval between the signal bits and reduce the impact of packet drops. As illustrated in Fig. 24, we know that the attacker is able to recognize the signal based on the size of sniffed IP packets using the signal detection mechanism discussed in Section IV-B in addition to using the cell count. In our recent work, we proposed a packet-size-based attack [38] that compromises Tor's communication anonymity with no need of controlling Tor routers. An attacker can manipulate size of packets between a Web site and an exit onion router and embeds a signal into the target traffic. An accomplice at the user side can sniff the traffic and recognize this signal. If the victim traffic is marked by our signal four times, the detection rate approaches over 90% with the delay interval of 400 ms, and the false positive rate can be suppressed to less than 4%.

VII. Related Work

A good review of mix systems can be found in [4] and [5]. There has been much research on degrading anonymous communication through mix networks. Existing traffic analysis attacks against anonymous communication can largely be categorized into two groups: passive traffic analysis and active watermarking techniques. Passive traffic analysis techniques have shown that the attacks record the traffic passively and identify the similarity between server's outbound traffic and client's inbound traffic [8], [9]. Other recent research works have shown that the attackers can infer sensitive information from the encrypted network traffic by examining patterns in terms of the sizes of packet and its timing [1], [39]–[41]. For example, Liberatore and Levine [40] examined the packet sizes of HTTP traffic transmitted over persistent connection or tunneled via SSH port forwarding can statistically identify the Web pages. Wright *et al.* [41] investigated the statistical distribution of packet sizes in encrypted Voice over IP (VoIP) connections

and identified the language spoken based on the distribution in each conversation. Later work of Wright *et al.* [42] also investigated how an eavesdropper could identify spoken phrases in encrypted VoIP.

The active watermarking techniques intend to embed specific secret signal (or marks) into the target traffic [10], [13], [17], [43]. Such techniques can reduce the false positive rate significantly if the signal is long enough and does not require massive training study of traffic cross correlation as required in passive traffic analysis. For example, Yu *et al.* [13] proposed a flow-marking scheme based on the DSSS technique. This approach could be used by attackers to secretly confirm the communication relationship via mix networks. Øverlier *et al.* [3] studied a scheme using one compromised mix router to identify the "hidden server" anonymized by Tor. Wang *et al.* [17] also investigated the feasibility of a timing-based watermarking scheme in identifying the encrypted peer-to-peer VoIP calls. Peng *et al.* [44] analyzed the secrecy of timing-based watermarking traceback proposed in [43], based on the distribution of traffic timing. Kiyavash *et al.* [37] proposed a multiflow approach detecting the interval-based watermarks [12], [45] and DSSS-based watermarks [13]. This multiflow-based approach intends to average the rate of multiple synchronized watermarked flows and expects to observe a unusual long silence period without packets or a unusual long period of low-rate traffic.

VIII. Conclusion

In this paper, we introduced a novel cell-counting-based attack against Tor. This attack is difficult to detect and is able to quickly and accurately confirm the anonymous communication relationship among users on Tor. An attacker at the malicious exit onion router slightly manipulates the transmission of cells from a target TCP stream and embeds a secret signal (a series of binary bits) into the cell counter variation of the TCP stream. An accomplice of the attacker at the entry onion router recognizes the embedded signal using our developed recovery algorithms and links the communication relationship among users. Our theoretical analysis shows that the detection rate is a monotonously increasing function with respect to the delay interval and is a monotonously decreasing function of the variance of one way transmission delay along a circuit. Via extensive real-world experiments on Tor, the effectiveness and feasibility of the attack is validated. Our data showed that this attack could drastically and quickly degrade the anonymity service that Tor provides. Due to Tor's fundamental design, defending against this attack remains a very challenging task that we will investigate in our future research.

Appendix A

Algorithm 1 shows the signal recovery mechanism with continuously embedded bits at a malicious Tor entry node. Algorithm 2 gives the signal recovery mechanism at a malicious Tor entry node when the time-hopping-based approach is used for embedding a signal into the target traffic.

Algorithm 1: Recovery Mechanism for Continuously Embedded Bits

Require:

(a) $C[1 * m]$, an array storing the number of cell counter variation in the circuit queue at the entry router;
(b) $S[1 * n]$, an array storing the original signal bit;

```
1: i = 0; j = 0
2: while i ≤ m do
3:    if C[i] == S[j] then
4:       Signal S[j] is matched.
5:    else if C[i] < S[j] then
6:       Signal S[j] is splitted.
7:       if C[i] + C[i + 1] == S[j] then
8:          Signal S[j] is processed as Type I with k = 1.
9:       else if C[i] + C[i + 1] > S[j] then
10:         Signal S[j] and S[j + 1] are processed as Type II
            with k = 1.
11:      else if C[i] + C[i + 1] < S[j] then
12:         Find the value of k
13:         if C[i] + ... + C[i + k] == S[j] then
14:            Signal S[j] is processed as Type I with k ≥ 2.
15:         else
16:            Signal S[j] and S[j+1] is processed as Type II
               with k ≥ 2.
17:         end if
18:         i = i + k;
19:      end if
20:      else if C[i] > S[j] then
21:      Two or more signals are combined together.
22:      if C[i] == S[j] + S[j + 1] then
23:         Signal S[j] and S[j + 1] are processed as Type II
            with k = 1.
24:      else if C[i] < S[j] + S[j + 1] then
25:         Signal S[j] and S[j+1] are processed as Type IV
            with k = 1.
26:      else if C[i] > S[j] + S[j + 1] then
27:         Find the value of k
28:         if C[i] == S[j] + ... + S[j + k] then
29:            These combined signals are processed as
               Type III with k ≥ 2.
30:         else
31:            These combined signals are processed as
               Type IV with k ≥ 2.
32:         end if
33:         j = j + k
34:      end if
35:   end if
36:   i = i + 1; j = j + 1
37: end while
```

Algorithm 2: Recovery Mechanism for Hopping-Based Encoding

Require

(a) $C[1 * m]$, an array storing the number of cell counter variation in the circuit queue at the entry router;
(b) $S[1 * n]$, an array storing the original signal bit;
(c) $Q[1 * n]$, an array storing the number of nonwatermark cells.

```
1: i = 0; j = 0
2: while i ≤ m do
3:    Remove the nonwatermark packets Q[j] from C[i].
4:       while Q[j] > C[i] do
5:       C[i + 1] = C[i + 1] + C[i]
```

end while
if $Q[j] == C[i]$ **then**
  $i = i + 1$; $Q[j]$ is removed.
  Detect $S[j]$ with C[i] by using Algorithm 1
else if $Q[j] < C[i]$ **then**
  The signal $S[j]$ is combined with $Q[j]$.
  $C[i] = C[i] - Q[j]$
  Detect $S[j]$ with $C[i]$ by using Algorithm 1
end if
$i = i + 1$; $j = j + 1$
end while

APPENDIX B

We now prove the fact that if L is a random variable with a log-normal distribution, $M = L/2$ also has a log-normal distribution. Let M be the random variable of one way delay through the circuit and M is approximately $L/2$, where L is a round-trip delay along the circuit. Since the CDF of one-way delay M is derived by

$$F_M(m) = P(M < m) \tag{34}$$
$$= P(L/2 < m) \tag{35}$$
$$= P(L < 2m) \tag{36}$$
$$= F_L(2m) \tag{37}$$
$$= \int_0^{2m} f(l)dl \tag{38}$$

the PDF of M can be derived by

$$f_M(m) = F'_M(m) \tag{39}$$
$$= F'_L(2m) \tag{40}$$
$$= 2f_L(2m) \tag{41}$$
$$= 2\frac{1}{2m\sigma_{x_i}\sqrt{2\pi}} e^{\frac{-(ln(2m)-\mu_{x_i})^2}{2\sigma^2_{x_i}}} \tag{42}$$
$$= \frac{1}{m\sigma_{x_i}\sqrt{2\pi}} e^{\frac{-(ln(m)-(\mu_{x_i}-ln2))^2}{2\sigma^2_{x_i}}}. \tag{43}$$

As we can see, the PDF of M follows a log-normal distribution as well.

REFERENCES

[1] Q. X. Sun, D. R. Simon, Y. Wang, W. Russell, V. N. Padmanabhan, and L. L. Qiu, "Statistical identification of encrypted Web browsing traffic," in *Proc. IEEE S&P*, May 2002, pp. 19–30.

[2] X. Fu, Y. Zhu, B. Graham, R. Bettati, and W. Zhao, "On flow marking attacks in wireless anonymous communication networks," in *Proc. IEEE ICDCS*, Apr. 2005, pp. 493–503.

[3] L. Øverlier and P. Syverson, "Locating hidden servers," in *Proc. IEEE S&P*, May 2006, pp. 100–114.

[4] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *Proc. IEEE S&P*, May 2003, pp. 2–15.

[5] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proc. 13th USENIX Security Symp.*, Aug. 2004, p. 21.

[6] "Anonymizer, Inc.," 2009 [Online]. Available: http://www.anonymizer.com/

[7] A. Serjantov and P. Sewell, "Passive attack analysis for connection-based anonymity systems," in *Proc. ESORICS*, Oct. 2003, pp. 116–131.

[8] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency MIX systems," in *Proc. FC*, Feb. 2004, pp. 251–565.

[9] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in Mix networks," in *Proc. PET*, May 2004, pp. 735–742.

[10] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proc. IEEE S&P*, May 2006, pp. 183–195.

[11] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against anonymous systems," in *Proc. ACM WPES*, Oct. 2007, pp. 11–20.

[12] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in *Proc. IEEE S&P*, May 2007, pp. 116–130.

[13] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "DSSS-based flow marking technique for invisible traceback," in *Proc. IEEE S&P*, May 2007, pp. 18–32.

[14] N. B. Amir Houmansadr and N. Kiyavash, "RAINBOW: A robust and invisible non-blind watermark for network flows," in *Proc. 16th NDSS*, Feb. 2009, pp. 1–13.

[15] V. Shmatikov and M.-H. Wang, "Timing analysis in low-latency MIX networks: Attacks and defenses," in *Proc. ESORICS*, 2006, pp. 18–31.

[16] V. Fusenig, E. Staab, U. Sorger, and T. Engel, "Slotted packet counting attacks on anonymity protocols," in *Proc. AISC*, 2009, pp. 53–60.

[17] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer VoIP calls on the internet," in *Proc. 12th ACM CCS*, Nov. 2005, pp. 81–91.

[18] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against anonymous systems," Univ. Colorado Boulder, Boulder, CO, Tech. Rep., Aug. 2007.

[19] X. Fu, Z. Ling, J. Luo, W. Yu, W. Jia, and W. Zhao, "One cell is enough to break Tor's anonymity," in *Proc. Black Hat DC*, Feb. 2009 [Online]. Available: http://www.blackhat.com/presentations/bh-dc-09/Fu/BlackHat-DC-09-Fu-Break-Tors-Anonymity.pdf

[20] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: Anonymity online," 2008 [Online]. Available: http://tor.eff.org/index.html.en

[21] R. Dingledine and N. Mathewson, "Tor protocol specification," 2008 [Online]. Available: https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=tor-spec.txt

[22] J. Reardon, "Improving Tor using a TCP-over-DTLS tunnel," Master's thesis, University of Waterloo, Waterloo, ON, Canada, Sep. 2008.

[23] R. Dingledine and N. Mathewson, "Tor path specification," 2008 [Online]. Available: https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt

[24] X. Fu, Z. Ling, W. Yu, and J. Luo, "Network forensics through cloud computing," in *Proc. 1st ICDCS-SPCC*, Jun. 2010, pp. 26–31.

[25] M. Perry, "TorFlow: Tor network analysis," in *Proc. 2nd HotPETs*, 2009, pp. 1–14.

[26] R. Pries, W. Yu, S. Graham, and X. Fu, "On performance bottleneck of anonymous communication networks," in *Proc. 22nd IEEE IPDPS*, Apr. 14–28, 2008, pp. 1–11.

[27] G. Smillie, *Analogue, Digital Communication Techniques*. London, U.K.: Butterworth-Heinemann, 1999.

[28] N. S. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on Tor using long paths," in *Proc. 18th USENIX Security Symp.*, Aug. 10–14, 2009, pp. 33–50.

[29] S. J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," in *Proc. 13th ACM CCS*, Nov. 2006, pp. 27–36.

[30] R. Pries, W. Yu, X. Fu, and W. Zhao, "A new replay attack against anonymous communication networks," in *Proc. IEEE ICC*, May 19–23, 2008, pp. 1578–1582.

[31] D. Mccoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the Tor network," in *Proc. 8th PETS*, 2008, pp. 63–76.

[32] S. U. Khaunte and J. O. Limb, "Packet-level traffic measurements from a Tier-1 IP backbone," Georgia Institute of Technology, Atlanta, GA, Tech. Rep., 1997.

[33] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley-Interscience, 1991.

[34] S. Verdu, "On channel capacity per unit cost," *IEEE Trans. Inf. Theory*, vol. 36, no. 5, pp. 1019–1030, Nov. 1990.

[35] "Tor: Anonymity online," The Tor Project, Inc., 2008 [Online]. Available: http://tor.eff.org/

[36] "PlanetLab | An open platform for developing, deploying, and accessing planetary-scale services," PlanetLab, 2011 [Online]. Available: http://www.planet-lab.org/

[37] N. Kiyavash, A. Houmansadr, and N. Borisov, "Multi-flow attacks against network flow watermarking schemes," in *Proc. USENIX Security Symp.*, 2008, pp. 307–320.

[38] Z. Ling, J. Luo, W. Yu, and X. Fu, "Equal-sized cells mean equal-sized packets in Tor?," in *Proc. IEEE ICC*, Jun. 2011, pp. 1–6.
[39] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on SSH," in *Proc. 10th USENIX Security Symp.*, Aug. 2001, p. 25.
[40] M. Liberatore and B. N. Levine, "Inferring the source of encrypted HTTP connections," in *Proc. ACM CCS*, Oct. 2006, pp. 255–263.
[41] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson, "Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob?," in *Proc. 16th Annu. USENIX Security Symp.*, Aug. 2007, pp. 43–54.
[42] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversation," in *Proc. IEEE S&P*, May 2008, pp. 35–49.
[43] X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter-packet delays," in *Proc. ACM CCS*, Nov. 2003, pp. 20–29.
[44] P. Peng, P. Ning, and D. S. Reeves, "On the secrecy of timing-based active watermarking trace-back techniques," in *Proc. IEEE S&P*, May 2006, pp. 335–349.
[45] Y. J. Pyun, Y. H. Park, X. Wang, D. S. Reeves, and P. Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *Proc. IEEE INFOCOM*, May 2007, pp. 634–642.

Zhen Ling received the B.S. degree in computer science from Nanjing Institute of Technology, Nanjing, China, in 2005, and is currently pursuing the Ph.D. degree in computer science and engineering at Southeast University, Nanjing, China.

He joined Department of Computer Science, City University of Hong Kong, Hong Kong, from 2008 to 2009 as a Research Associate, and then joined the Department of Computer Science, University of Victoria, Victoria, BC, Canada, in 2011 as a visiting scholar. His research interests include network security, privacy, and forensics.

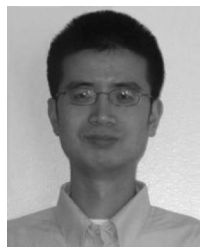
Junzhou Luo (M'10) received the B.S. degree in applied mathematics and M.S. and Ph.D. degrees in computer network from Southeast University, Nanjing, China, in 1982, 1992, and in 2000, respectively.

He is a Full Professor with the School of Computer Science and Engineering, Southeast University. His research interests are next-generation network, protocol engineering, network security and management, grid and cloud computing, and wireless LAN.

Prof. Luo is Co-Chair of the IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design.

Wei Yu received the B.S. degree in electrical engineering from Nanjing University of Technology, Nanjing, China, in 1992, the M.S. degree in electrical engineering from Tongji University, Shanghai, China, in 1995, and the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2008.

He is an Assistant Professor with the Department of Computer and Information Sciences, Towson University, Towson, MD. Before that, he worked for Cisco Systems, Inc., San Jose, CA, for almost nine years. His research interests include cyberspace security, computer network, and distributed systems.



Xinwen Fu received the B.S. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1995, the M.S. degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 1998, and the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2005.

He is an Assistant Professor with the Department of Computer Science, University of Massachusetts Lowell, Lowell, which he joined in the summer of 2008 as a faculty member. From 2005 to 2008, he was an Assistant Professor with the College of Business and Information Systems, Dakota State University, Madison, SD. His current research interests are in network security and privacy.

Dong Xuan received the B.S. and M.S. degrees in electronic engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 1990 and 1993, respectively, and the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2001.

Currently, he is an Associate Professor with the Department of Computer Science and Engineering, The Ohio State University (OSU), Columbia. He was on the faculty of Electronic Engineering at SJTU from 1993 to 1998. His research interests include distributed computing, computer networks, and cyberspace security.

Dr. Xuan received the NSF CAREER Award in 2005 and the Lumley Research Award from the College of Engineering, OSU, in 2009.

Weijia Jia received the B.Sc. and M.Sc. degrees from Center South University, Changsha, China, in 1982 and 1984, respectively, and the Master of Applied Science and Ph.D. degrees from the Polytechnic Faculty of Mons, Mons, Belgium, in 1992 and 1993, respectively, all in computer science.

He is currently a Full Professor with the Department of Computer Science and the Director of Future Networking Center, ShenZhen Research Institute, City University of Hong Kong (CityU), Hong Kong. He joined the German National Research Center for Information Science (GMD), Bonn (St. Augustine), Germany, from 1993 to 1995 as a Research Fellow. In 1995, he joined the Department of Computer Science, CityU, as an Assistant Professor. His research interests include next-generation wireless communication, protocols and heterogeneous networks; distributed systems, and multicast and anycast QoS routing protocols.