A New Cell Counter Based Attack Against Tor

Zhen Ling[*¶], Junzhou Luo[*], Wei Yu[†], Xinwen Fu[‡], Dong Xuan[§], Weijia Jia[¶]

[*]Southeast University
2 Sipailou,
Nanjing 210096, P.R.China
{zhen_ling, jluo}@seu.edu.cn

[†]Towson University
8000 York Road
Towson, MD 21252, USA
weyu@cisco.com

[‡]University of Massachusetts Lowell
One University Avenue,
Lowell, MA 01854, USA
xinwenfu@cs.uml.edu

[§]The Ohio State University
2015 Neil Avenue,
Columbus, OH 43210, USA
xuan@cse.ohio-state.edu

[¶]City University of Hong Kong
83 Tat Chee Avenue,
Kowloon, Hong Kong SAR, China
{zhenling, wei.jia}@cityu.edu.hk

ABSTRACT

Various low-latency anonymous communication systems such as Tor and Anoymizer have been designed to provide anonymity service for users. In order to hide the communication of users, many anonymity systems pack the application data into equal-sized cells (e.g., 512 bytes for Tor, a known real-world, circuit-based low-latency anonymous communication network). In this paper, we investigate a new cell counter based attack against Tor, which allows the attacker to confirm anonymous communication relationship among users very quickly. In this attack, by marginally varying the counter 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).

Categories and Subject Descriptors

C.2.0 [**Computer Network**]: General—*Security and Protection*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*; E.3 [**Data**]: Encryption

CCS'09, November 9–13, 2009, Chicago, Illinois, USA.

General Terms

Security, Reliability

Keywords

Cell Counter, Signal, Anonymity, Mix Networks, Tor

1. 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 (LBS), and E-voting. In these applications, encryption alone cannot maintain the anonymity required by participants [32, 15, 24]. Since Chaum pioneered the basic idea of anonymous communication techniques, referred to as mixes [7], 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. Email is a typical message-based anonymity application, which has been thoroughly investigated [8, 16]. 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 [30, 12, 2].

To degrade the anonymity service provided by anonymous communication systems, traffic analysis attacks have been studied [24, 19, 40, 23, 4, 36, 39, 1]. 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 sender's outbound traffic and receiver's inbound traffic based on statistical measures [40, 19]. This type of technique requires a long period of observation for the traffic in order to reduce errors of attack. 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 [37, 39].

In this paper, we focus on the active watermarking technique, which has been active in the past few years. For ex-

ample, Wang *et al.* [37] investigated a timing watermarking scheme to identify the encrypted peer-to-peer VoIP calls. The idea of this scheme is to change the timing interval of packets from Alice to Bob and then correlate the timing interval at packets received by Bob. Nevertheless, this timing-based scheme is not effective to confirm the anonymous communication sessions through a mix network which deploys batching strategies to manipulate the timing interval of inter-packet delivery. Yu *et al.* [39] 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 changes. All these make it hard for the flow marking technique to deal with the short communication sessions which may last for 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 fastness of confirming anonymous communication relationships among users. Although accuracy and/or detectability have received great attention [39, 37], to the best of our knowledge, no existing work can meet all three requirements simultaneously.

In this paper, we investigate a new cell counter based attack against Tor, a real-world, circuit-based low-latency anonymous communication network. This attack 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 not 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 cell counter in the target traffic. An accomplice of the attacker at the entry onion router detects and excludes the *control* cells, records the counter 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.

We have implemented the cell counter 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 great impact on Tor, because the attack can confirm over half of communication sessions by injecting around 10% malicious onion routes on Tor [5, 14].

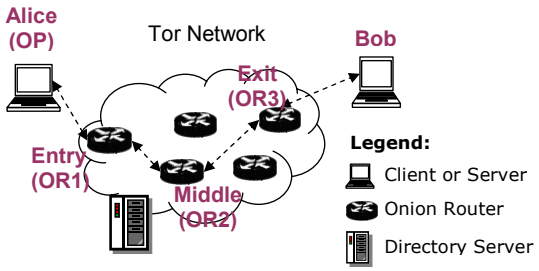

Figure 1: Tor Network

The remainder of this paper is organized as follows: We introduce the background in Section 2. We present the cell counter based attack, including the basic idea, issues of the attack and solutions in Section 3. At the end of Section 3, we discuss the extension, detectability and impact of the proposed attack. In Section 4, we analyze the effectiveness of the attack. In Section 5, we show experimental results on Tor and validate our findings. We review related work in Section 6 and conclude this paper in Section 7.

2. BACKGROUND

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

2.1 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 [11]. Figure 1 illustrates the basic components of Tor [10]. As shown in Figure 1, there are four basic components: (i) *Alice* (i.e. *Client*). The client runs a local software called *onion proxy* (*OP*) to anonymize the client data into Tor. (ii) *Bob* (i.e. *Server*). It runs TCP applications such as a web service and anonymously communicates with *Alice* over Tor. (iii) *Onion routers (OR).* 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 bytes as shown in Figure 2) carried through TLS connections. (iv) *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.

Figure 2 illustrates the cell format used by Tor. All cells have a three-bytes header, which is not encrypted in the onion-like fashion so that the intermediate Tor routers can see this header. The other 509 bytes are encrypted in the onion-like fashion. There are two types of cells: *control* cell shown in Figure 2 (a) and *relay* cell shown in Figure 2 (b).

2.2 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 choosing several *OR*s (preferably ones with high-

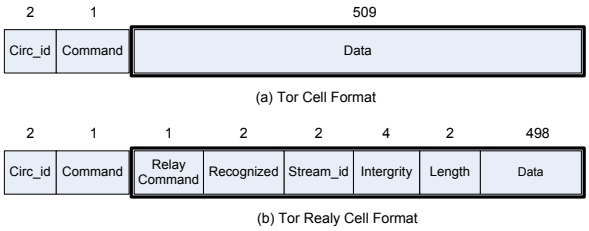

Figure 2: Cell Format By Tor

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 below 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 handle the TCP streams from client applications. Then the *OR* on the other side of the circuit connects to the requested destinations and relays the data.

Figure 3 illustrates the procedure that the *OP* establishes a circuit and downloads a file from the server. In Figure 3, *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 [10]. In this way, a one-hop circuit *C1* is created. Similarly, the *OP* extends the circuit to the 2-hop circuit and 3-hop circuit.

After the circuit is setup 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.

2.3 Processing Cells at Onion Routers

Figure 4 illustrates the procedures of processing cells at onion routers. Notice 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 at 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 bytes. Thus, 512 bytes data will be pulled out from the input buffer every time until the data remaining in the connection input buffer is small than 512 bytes. Since each onion router has a routing table that maintains the map

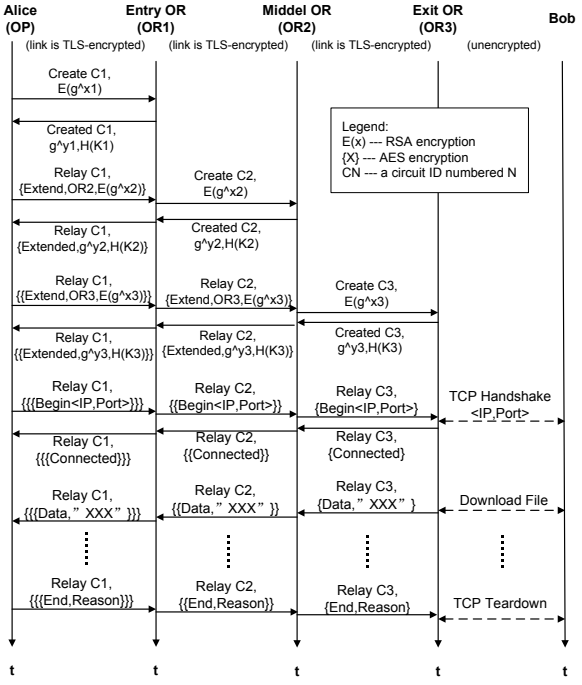

Figure 3: Circuit Creation and Data Transmission on Tor

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 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. Therefore, the 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 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.

3. CELL COUNTER BASED ATTACK

In this section, we first introduce the basic idea of the cell counter based attack against Tor. We then list some challenging issues related to the attack and present solutions to address those issues. At the end, we discuss the extension, detectability and impact of the proposed attack.

3.1 Basic Idea

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

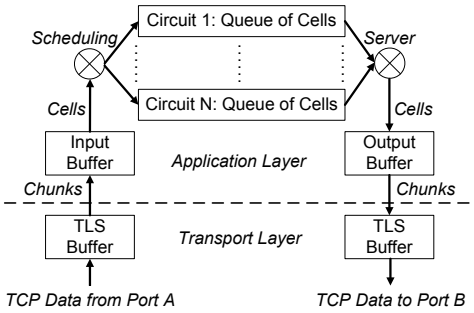

Figure 4: Processing the Cells at Onion Routers

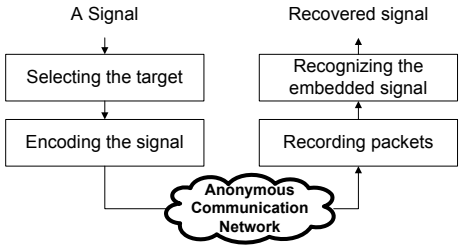

Figure 5: Workflow of Cell Counter Based Attack

small percentage of exit and entry onion routers by donating computers to Tor, similar to other studies [23, 24, 14, 5]. The assumption is valid, since Tor is operated in a voluntary manner [10]. 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 counter 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 transmitted 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 signal is recognized, the attacker confirms the communication relationship between Alice and Bob. As shown in Figure 5, the workflow of the cell counter based attack is illustrated below.

Step 1: Selecting the target. At a malicious exit onion router connected to server Bob, the attacker will log the information, including the 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 stream data. According to the description of Tor in Section 2, 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 a *CELL_RELAY_DATA* cell and the attacker enters the encoding process shown in Step 2.

Step 2: Encoding the signal. In Section 2, we described the procedures of how to process cells at the onion routers. The *CELL_RELAY_DATA* cells will be waiting in the circuit queue at 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 counter in the target traffic. Particularly, in order to encode the "1" bit of original signal, the attacker lets three cells to be flushed from the queue. In order to encode the "0" bit of the signal, the attacker lets only one cell to be flushed from the queue. In order to accurately manipulate the number of the cells to the circuit queue, 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 lets the circuit write event to be called 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 developed reliable encoding mechanisms to combat network dynamics in Section 3.2.

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 along with 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 cell counters 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 arrives at the entry onion router are *CELL_RELAY_EXTENDED* cell and the third cell is *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 3.2 to decode the embedded signal. Once the original signal is identified, since the entry onion router knows the Alice's host IP address and the exit onion router knows Bob's host IP address of the TCP stream, 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 the three cells for encoding the "1" bit of the signal arrives 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. Moreover, 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 the "1" bit of the signal may be changed to two portions. The first portion contains the first cell and the second portion contains the second and third cell together. Therefore, care must be paid to take this into account in order to recognize the "1" bit of the signal in the case above. 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 [29]. All these facts cause the distorted version of the originally embedded signal to be re-

ceived at the entry onion router. To deal with this issue, we designed mechanisms to carefully encode and robustly recover the embedded signal in Section 3.2.

3.2 Issues and Solutions

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

3.2.1 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 counter of cells and embed the secret signal into the variation of cell counters in the target traffic. If the attacker uses two cells to encode the "1" bit of the signal, the "1" bit will be easily distorted over the network and will be hard to recover. The reason can be explained as follows: when the two cells arrive at the input buffer at the middle onion router, the first cell will be pulled to the circuit queue. If the output buffer is empty, the first cell will be flushed to 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 to be called. Consequently, two originally combined cells will be divided 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. Therefore, these two cells will be decoded as two "0" bits of the signal, leading to an inaccurate detection of the signal. To deal with this problem, the attacker should choose at least three cells for carrying "1" bit of the signal. If the middle onion router splits them into one cell and two cells, the attacker can still recognize them as "1" bit of the signal at the entry onion router.

Proper delay interval should be selected for transmitting cells. Since the signal modulates the counter of cells transmitted from the exit onion router to the entry onion router, the delay interval 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 selected to be 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's use one simple example to clarify this. We assume that the delay interval 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 calling 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 3.2.2.

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 ($j \in [0, n-1)$]) be 0 or 1. When $S_j = 1$, the attacker will choose three cells to encode the bit "1". When $S_j = 0$, the attacker will choose only one cell to encode the 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 can pull 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 that the attacker modulates cells carrying two sequential bits of the signal be I and let the delay of the data transmitted in the network between *OR3* and *OR2* be D. Therefore, the relationship between T_i and T_{i+1} can be represented by,

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

Let the time of the cells for the signal S that arrives at the circuit queue be T^{queue}, where $T_i^{queue} = T_i + T_{read}$. Let the time of the cells for the signal S that arrives at the output buffer as $T^{outbuffer}$, where $T_i^{outbuffer} = T_i + T_{read} + T_{write}$[1].

In order to avoid the combination of cells which 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

$$\begin{aligned} T_i^{outbuffer} &\leq T_{i+1}^{queue}, & (2)\\ T_i + T_{read} + T_{write} &\leq T_{i+1} + T_{read}, & (3)\\ T_i + T_{write} &\leq T_i + I + D, & (4)\\ T_{write} &\leq I + D. & (5) \end{aligned}$$

The parameter T_{write} is affected by the network condition. Suppose that the network is congested, i.e., $T_{write} > I + D$, the write event in the event queue cannot be called on time to flush the cells in the output buffer and the circuit queue. Then the following 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, T_{write} is small, i.e., $T_{write} < I + D$. The cells will be transmitted on 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., $T_{write} \approx I + D$, 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.

[1]Please refer to [29] for statistics of T_{read}, T_{write} and other related random variables.

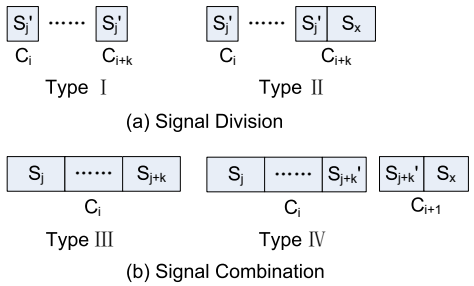

Figure 6: Signal Division and Combination

3.2.2 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. Figure 6 (a) illustrates two types of the cell division for the unit of the signal, and Figure 6 (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 counters 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. According to the notation, the original signal is denoted as $S = \{S_0, S_1, \ldots, S_j, \ldots, S_{n-1}\}$. Let S_j be the *j-th* signal, $S_j^{'}$ as the part of the *j-th* signal, and let S_x be the remaining signal in the packet or a null signal. Type I of the signals indicates that the original signal S_j is divided into $k+1$ separate cells. Figure 7 illustrates an example for Type I with $k = 1$. Suppose signal S_j is "1" bit, the number of the 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 of the signals indicates that the last part of the S_j is merged with the following signal(s) S_x. Figure 7 illustrates an example for Type II with $k = 1$. Suppose signal S_j is "1" bit and S_x is a whole 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 of the signals indicates that k original signal are merged into a signal packet. Figure 7 illustrates an example for Type III with $k = 2$. If signal S_j, S_{j+1} and S_{j+2} are "010" bits, the attacker records that C_i is 5. In this case, the cells belonging to three signal units are merged all together. Type IV of the signals indicates that the part of the S_{j+k} is divided into the following cells. Figure 7 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. In the above, we only give simple examples of four types. Due to diverse network condition, the division or combination of the cells in these types may be even more complicated.

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 the cells recorded in the circuit queue is smaller than the number of the cells of the original signal, the signals are recovered as either Type I or Type II. Suppose the counter of the cells recorded in the circuit queue is larger than the counter of the 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.

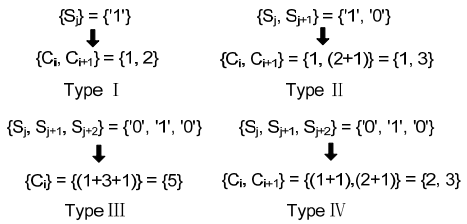

Figure 7: Examples of Signal Division and Combination

3.3 Discussion

In this subsection, we discuss how an adversary uses only Tor exit routers for the attack, attack detectability and impact.

3.3.1 Controlling Exit Onion Routers Only

If the attacker does not control the entry onion routers, the cell counter based attack can still be successful with some alternatives, e.g., sniffing the packets transmitted between an entry onion router and a client. According to the length of packets sniffed, the attacker can detect the embedded signal based on the size of the packet. Without loss of generality, we assume that the MTU (Maximum Transmission Unit) is 1500 bytes.

We now introduce the structure of the IP packet that envelops the cell(s) and passes along the network. Figure 8 (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 bytes), a TLS message (not to exceed 2^{14} bytes), a MAC (Message Authentication Code, 20 bytes) and a TLS padding (12 bytes). Figure 9 illustrates the header of the TLS packet, with the length of five bytes. The field of content type identifies the record layer protocol type contained in this record, with the length of one byte. In our case, we concern the 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 two bytes. The field of length identifies the length of protocol message(s), not to exceed 2^{14} bytes.

Figure 8 (b) illustrates the structure of IP packet that envelops two cells which has a length of 1150 bytes. Because an IP packet that envelops three cells exceeds the MTU (1500 bytes), this IP packet will be segmented; one segment has the packet with 1500 bytes and the other segment has the packet with 214 bytes. Figure 8 (c) illustrates the structure of IP packet that envelops three cells and is segmented. Therefore, the attacker can map "0" bit of the signal to one IP packet, with the length of 638 bytes. 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 Figure 8 (a)) and two cells (shown in Figure 8 (b)), i.e., the signal is divided as Type I with $k = 1$, as well as two IP packets with three cells (shown in Figure 8 (c)) which is neither divided nor combined. As such, the attacker is still able to recognize the signal based on the size

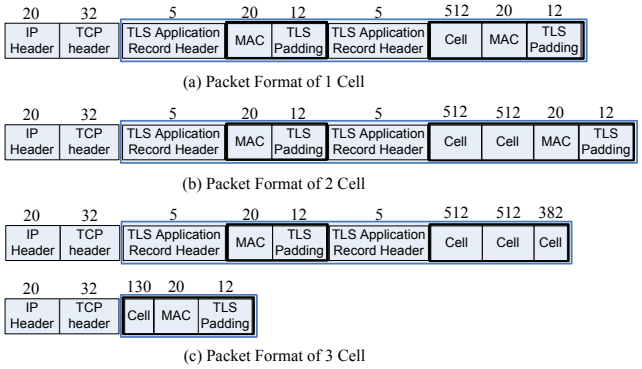

Figure 8: Packet Format

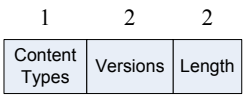

Figure 9: TLS header

of sniffed IP packets using the signal detection mechanism discussed above.

The fact that multiple cells can be packed into a packet guarantees the correct signal encoding via the cell counter. 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 unit "1" or "0".

3.3.2 Attack Detectability

The proposed cell counter-based attack is difficult to detect for the following reasons. First, the attack transmits a short and secret random signal known only to the attackers. It is difficult to detect within the target traffic. According to Figure 12, the success of this attack requires only a short secret signal-such as 5 bits-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.

Second, we adopt time hopping-based signal embedding techniques to further improve the attack detectability. The interval between two bits is controlled by a secret pseudonoise code and known only to attackers. Intuitively, if the interval between bits is large enough, the inserted signal bits appear sparse within the target traffic and it is difficult to determine if groups of cells are caused by network dynamics or by intention. Therefore, the secret signal embedded into the target traffic is no different than noise. Moreover, when a malicious entry node has confirmed the communication relationship, it can separate the group of cells by adding delay between cells so that not even the client can observe the embedded signal. In Section 5, we demonstrate the effectiveness of this time hopping-based technique and the detailed approach is presented as 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 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 5's figures imply that normal network traffic does have groups of one and three cells caused by network dynamics. Moreover, 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.

3.3.3 Attack Impacts

The proposed cell counter-based attack may dramatically degrade the anonymity that the Tor network maintains. Unlike other existing attacks, the 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 [37, 39, 23, 13, 22]. 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 [26, 14] 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 and correlate the counting. However, there is no guarantee of detection rate and false positive rate because of the huge 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 5. Therefore, as a powerful traffic confirmation attack, the proposed attack poses a great challenge against Tor. In our recent study [14], we showed that by compromising around 10% of onion routers, the attack can confirm over half of the communication sessions over Tor. The attacker can donate malicious onion routers with long uptime and high bandwidth into the Tor network in order to increase the probability that malicious onion routers are selected for either an entry or exit router [10]. Our results were consistent with the observations from Bauer *et al.* [5], which indicate that the attacker can compromise approximately 46.46% circuits with 9% malicious routers within Tor[2].

It is also difficult for Tor to defeat the proposed attack. One possible way is to let Tor ORs add delays between cells to disrupt malicious cell groups. However, choosing this delay will be very challenging. Too short a delay cannot separate cells at lower layers, whereas a relatively large delay may dramatically degrade Tor's performance, which is already the biggest bottleneck of using Tor [27, 29, 21]. 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 hitting malicious Tor ORs. However, a random routing algorithm will also degrade Tor performance.

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. Indeed, our attack exploits the Tor protocol's impact on the network layer. Note that we have also taken a preliminary look at the design and implementation of Anonymizer [2] (another popular anonymous communication system) and observed a similar problem. We were able to control the size of packets passing through Anonymizer. This type of attack may actually be applied against any system that relies on packet size to preserve privacy.

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

4. ANALYSIS

In this section, we show the analytical results for the accuracy and efficiency of the cell counter based attack. For the attack accuracy, we derive the closed formula to show the detection rate and false positive rate.

4.1 Detection Rate

The round-trip delay between two onion routers can be modeled by a lognormal distribution [17]. Notice that our analysis is based on the network shown in Figure 3. A lognormal random variable has the property - its logarithm has a Gaussian distribution. Let X_i be a Gaussian random variable with the probability density function (PDF),

$$f_{x_i}(x) = \frac{1}{\sigma_{x_i}\sqrt{2\pi}} e^{-\frac{(x-\mu_{x_i})^2}{2\sigma_{x_i}^2}}, \quad (6)$$

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 lognormal distribution and the PDF of L_i is given by

$$f_{L_i}(l) = \frac{1}{l\sigma_{x_i}\sqrt{2\pi}} e^{-\frac{(ln(l)-\mu_{x_i})^2}{2\sigma_{x_i}^2}}. \quad (7)$$

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

$$L = L_1 + L_2 = e^{X_1} + e^{X_2} \approx e^H, \quad (8)$$

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

Now we derive the detection error rate. Let n be the length of the original signal, 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 that the attacker transmits cells 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

$$\begin{aligned} P_e &= Pr(T_{s_i} > T_{s_{i+1}} + \Delta t), \\ &= Pr(T_{s_i} - T_{s_{i+1}} > \Delta t). \end{aligned} \quad (9)$$

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

$$\begin{aligned} P_e &= Pr(Z > \Delta t), \\ &= 1 - Pr(Z \le \Delta t). \end{aligned} \quad (10)$$

The detection rate P_D is defined as the probability that a 1-bit original signal is correctly recognized. We have

$$\begin{aligned} P_D &= 1 - P_e, \\ &= Pr(Z \le \Delta t). \end{aligned} \quad (11)$$

Let $T_{s_i} \doteq X$ and $T_{s_{i+1}} \doteq Y$. We have $Z = X - Y$. Because 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's ($\ln X$ or $\ln Y$). Since

$$\begin{aligned} F_Z(z) &= Pr(Z \le z) = Pr(X - Y \le z), \\ &= \int_0^{+\infty} (\int_0^{z+y} f(x,y)dx)dy, \end{aligned} \quad (12)$$

then

$$\begin{aligned} Pr(Z \le \Delta t) &= F_Z(\Delta t), \\ &= \int_0^{+\infty} (\int_0^{\Delta t+y} f(x,y)dx)dy, \\ &= \int_0^{+\infty} f(y)(\int_0^{\Delta t+y} f(x)dx)dy, \\ &= \int_0^{+\infty} f(y)(\frac{1}{2} + \\ &\quad \frac{1}{2}erf(\frac{\ln(\Delta t+y)-\mu}{\sigma\sqrt{2}}))dy, \end{aligned} \quad (13)$$

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

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

$$\begin{aligned} F_Z'(\Delta t) &= \frac{1}{2} \int_0^{+\infty} f(y) \frac{1}{(\Delta t+y)\sigma\sqrt{2}} \frac{2}{\sqrt{\pi}} e^{-(\frac{\ln(\Delta t+y)-\mu}{\sigma\sqrt{2}})^2} dy, \\ &= \frac{1}{\sigma\sqrt{2\pi}} \int_0^{+\infty} \frac{1}{\Delta t+y} e^{-(\frac{\ln(\Delta t+y)-\mu}{\sigma\sqrt{2}})^2} f(y)dy. \end{aligned} \quad (14)$$

Since $\Delta t > 0$ and $y > 0$, we have $F_Z'(\Delta t) > 0$. Hence, $P_D > 0$ and P_D is a monotone 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 5.

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

$$P_{D,n} = (P_D)^n, \quad (15)$$

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

4.2 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 the packet be p_0 (which will be recognized as "0" bit of original signal). Let the probability of three cells packed in the packet be p_1 (which will be recognized as "1" bit of original signal). Let p_2 be the probability that packets have other sizes. We have $p_2 = 1 - p_0 - p_1$.

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

$$P_{F,n} = (\frac{p_0+p_1}{2})^n = (\frac{1-p_2}{2})^n. \quad (16)$$

Figure 10 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

$$P_{F,n} \approx (\frac{0.5}{2})^n = (\frac{1}{4})^n. \quad (17)$$

As we can see, we will have lower false positive rates, as the original signal length n gets longer. Given the false positive rate $P_{F,n}$ in (17), 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 5, we observed even much lower false positive rate.

5. EVALUATION

We have implemented the cell counter based attack presented in Section 3 against Tor [33]. 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.

5.1 Experiment Setup

In our experiment setting, 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 [34]. The server (Bob) is located at one University campus in North American 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). Figure 11 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 Figure 11. The middle onion router is selected using the default routing selection algorithm released by Tor. As we stated earlier, the cell counter based attack intends to confirm whether the client (Alice) communicates with the server (Bob). For verification purposes, we setup 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* [9], we let the client choose both the malicious entry and exit onion routers along the circuit.

5.2 Experimental Results

To validate the accuracy of the cell counter based attack, we let the client download 30 files in our experiments. At the exit onion router, we generate a random signal with 100-bits. 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 cell counter 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 use the data to recognize the embedded signal. In addition, we chose different thresholds and types in our recovery mechanism. In particular, we chose to recover Type I and III with $k = 1$ 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-bit random signal and apply detection scheme 1 and detection scheme 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.

Figure 12 illustrates the correlation between detection rate (true positive) and 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 (the overall false positive rate is a fixed value). When the delay interval approaches 100ms, 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.

Figure 13 illustrates the detection rate in terms of signal length and the delay interval for scheme 1. 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 is constantly very low (less than 5%). When the signal length is 20, and the delay interval between signals is 100 ms, the 100% detection rate can be achieved. This validates that the investigated attack only requires tens of cells and is highly efficient to confirm very short communication sessions on Tor. Figure 14 illustrates the detection rate in terms of signal length and the delay interval for scheme 2. The false positive decreases quickly with the increasing signal length. Additionally, the detection rate can approach 100% with the delay interval 100 ms and signal length 100 with a low false positive.

To further improve the detectability of cell counter based attack, we also investigated an improved encoding mechanism, called the hopping-based encoding, that randomly embeds units of a signal into the target traffic. This improved encoding mechanism can resist the multi-flow attack [18]. In this new 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, the multi-flow attack is hard to detect the embedded signal in the traffic. Figure 15 illustrates relationship between detection rate (true positive) and the mean λ of non-watermarked cells length (which is the random time interval that no signal is embedded). From this figure, 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 in Appendix A, we use Algorithm 2 in Appendix A to remove the cells without carrying any unit of the signal and to detect the remaining signal.

6. RELATED WORK

A good review of mix systems can be found in [12, 8]. 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. The passive traffic analysis attacks record the traffic passively and identify the similarity between Alice's outbound traffic and Bob's inbound traffic. For example, Zhu *et al.* [40] proposed the scheme of using mutual information for the similarity measurement. Levine *et al.* [19] investigated a cross correlation technique for the similarity measurement.

The active watermarking techniques intend to embed special signal (or marks) into the target traffic, including traffic rate and delay interval between packets [37, 39]. For example, Yu *et al.* [39] proposed a flow marking scheme based on the direct sequence spread spectrum (DSSS) technique. This approach could be used by attackers to secretly confirm the communication relationship via mix networks. Murdoch *et al.* [23] also investigated the timing-based attacks

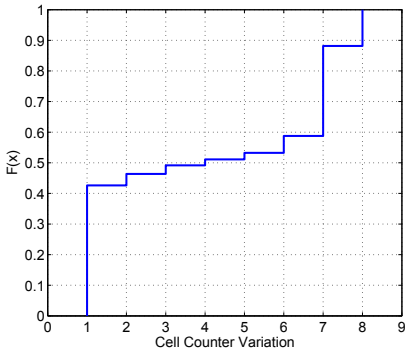

Figure 10: Cumulative Probability Function for Packet Size in Normal Traffic

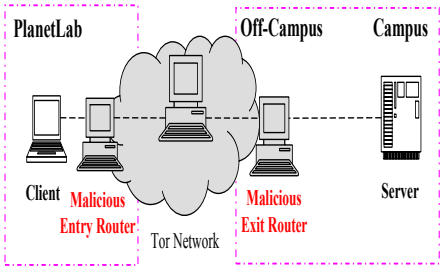

Figure 11: Experiment Setup

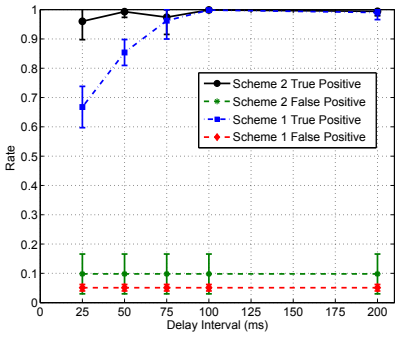

Figure 12: Detection Rate vs. Delay Interval (Note: the rate is for detecting one bit)

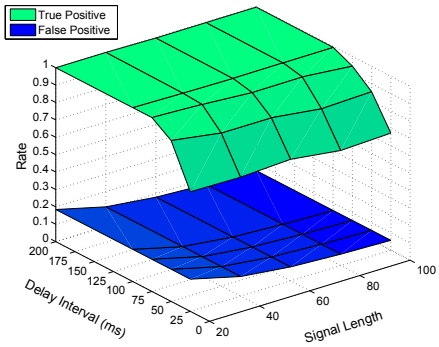

Figure 13: Detection Rate vs. Delay Interval and Signal Length with Detection Scheme 1 (Note: the rate is for detecting one bit)

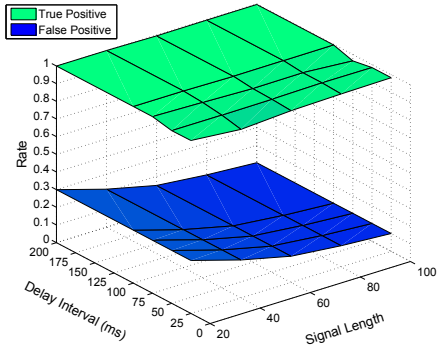

Figure 14: Detection Rate vs. Delay Interval and Signal Length with Detection Scheme 2 (Note: the rate is for detecting one bit)

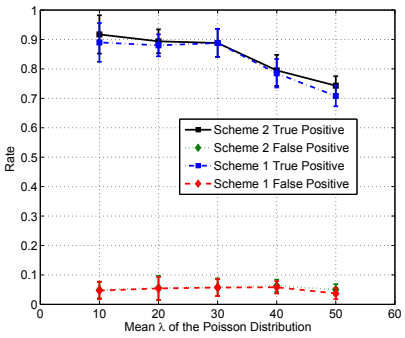

Figure 15: Correlation Between Detection Rate and Mean λ of the Poisson Distribution (Note: the rate is for detecting one bit)

on Tor by using some compromised Tor routers. Overlier *et al.* [24] studied a scheme using one compromised mix router to identify the "hidden server" anonymized by Tor. Wang *et al.* [38] proposed an active watermarking scheme that was robust to random timing perturbation. They analyzed the tradeoffs between the true positive rate, the maximum timing perturbation added by attackers, and the number of packets needed to successfully decode the watermark. Wang *et al.* [37] also investigated the feasibility of a timing-based watermarking scheme in identifying the encrypted peer-to-peer VoIP calls. By slightly changing the timing of packets, their approach can correlate encrypted network connections. Nevertheless, these timing-based schemes are not effective at tracing communication through a mix network with batching strategies that manipulate inter-packet delivery timing. Peng *et al.* [25] analyzed the secrecy of timing-based watermarking traceback proposed in [38], based on the distribution of traffic timing. Kiyavash *et al.* [18] proposed a multi-flow approach detecting the interval-based watermarks [28, 36] and DSSS-based watermarks [39]. This multi-flow 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.

Randomization has been the approach proposed to preserve privacy in various scenarios including data privacy [20] and watermarking [36]. Our study is also related to the covert channel. Various covert channels have been studied for different applications [6, 31, 3]. For example, JitterBugs is a class of inline interception mechanisms that covertly transmit data by perturbing the timing of input events in order to affect externally observable network traffic. Takahshi *et al.* in [35] assessed VoIP covert channel threats that utilize an IP phone conversation to illicitly transfer information across the network.

7. CONCLUSION

In this paper, we introduced a novel cell counter 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 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 link the communication relationship among users. Via extensive theoretical analysis and 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.

Acknowledgment

This work was supported in part by the National Science Foundation under grants 0721766, 0907964 and 0546668, by the Army Research Office (ARO) under grant No. AMSRD-ACC-R50521-CI, by CityU Applied R & D Centre (ARD(Ctr)) No. 9681001, RGC General Research Fund (GRF), SAR Hong Kong, No. 9041350 (CityU 114908), and Jiangsu Provincial Natural Science Foundation of China under Grant No. BK2007708 and BK2008030, China Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 200802860031, and by Jiangsu Provincial Key Laboratory of Network and Information Security under Grant No. BM2003201. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor agencies.

8. REFERENCES

[1] N. B. Amir Houmansadr, Negar Kiyavash. Rainbow: A robust and invisible non-blind watermark for network flows. In *Proceedings of the 16th Network and Distributed System Security Symposium (NDSS)*, February 2009.

[2] Anonymizer, Inc. `http://www.anonymizer.com/`, 2009.

[3] D. Bailey, D. Boneh, E.-J. Goh, and A. Juels. Covert channels in privacy-preserving identification systems. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security (CCS)*, November 2007.

[4] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against anonymous systems. In *Proceedings of ACM Workshop on Privacy in the Electronic Society (WPES)*, October 2007.

[5] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against anonymous systems. Technical report, University of Colorado at Boulder, August 2007.

[6] S. Cabuk, C. Brodley, and C. Shields. Ip covert timing channels: design and detection. In *Proceedings of the 2004 ACM Conference on Computer and Communications Security (CCS)*, October 2004.

[7] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[8] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: design of a type iii anonymous remailer protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (S&P)*, May 2003.

[9] R. Dingledine and N. Mathewson. Tor path specification. `http://tor.eff.org/svn/trunk/doc/spec/path-spec.txt`, 2008.

[10] R. Dingledine and N. Mathewson. Tor protocol specification. `http://tor.eff.org/svn/trunk/doc/spec/tor-spec.txt`, 2008.

[11] R. Dingledine, N. Mathewson, and P. Syverson. Tor: anonymity online. `http://tor.eff.org/index.html.en`, 2008.

[12] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[13] N. S. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on tor using long paths. In *Proceedings of the 18th USENIX Security Symposium*, August 10-14 2009.

[14] X. Fu, Z. Ling, J. Luo, W. Yu, W. Jia, and W. Zhao. One cell is enough to break tor's anonymity. In *Proceedings of Black Hat DC*, February 2009.

[15] X. Fu, Y. Zhu, B. Graham, R. Bettati, and W. Zhao. On flow marking attacks in wireless anonymous communication networks. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, April 2005.

[16] C. Gülcü and G. Tsudik. Mixing email with babel. In *Proceedings of the Network and Distributed Security Symposium (NDSS)*, February 1996.

[17] S. U. Khaunte and J. O. Limb. Packet-level traffic measurements from a tier-1 ip backbone. Technical report, Georgia Institute of Technology, 1997.

[18] N. Kiyavash, A. Houmansadr, and N. Borisov. Multi-flow attacks against network flow watermarking schemes. In *Proceedings of USENIX Security Symposium*, 2008.

[19] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright. Timing attacks in low-latency mix-based systems. In *Proceedings of Financial Cryptography (FC)*, February 2004.

[20] Y. J. Li, V. Swarup, and S. Jajodia. Fingerprinting relational databases: schemes and specialties. *IEEE Transactions on Dependable and Secure Computing*, 2(1), January 2005.

[21] D. McCoy, K. Bauer, D. Grunwald, P. Tabriz, and D. Sicker. Shining light in dark places: A study of anonymous network usage. Technical report, University of Colorado at Boulder, August 2007.

[22] S. J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, November 2006.

[23] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.

[24] L. Overlier and P. Syverson. Locating hidden servers. In *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.

[25] P. Peng, P. Ning, and D. S. Reeves. On the secrecy of timing-based active watermarking trace-back techniques. In *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.

[26] R. Pries, W. Yu, X. Fu, and W. Zhao. A new replay attack against anonymous communication networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, May 19-23 2008.

[27] R. Pries, W. Yu, S. Graham, and X. Fu. On performance bottleneck of anonymous communication networks. In *Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, April 14-28 2008.

[28] Y. J. Pyun, Y. H. Park, X. Wang, D. S. Reeves, and P. Ning. Tracing traffic through intermediate hosts that repacketize flows. In *Proceedings of IEEE INFOCOM*, May 2007.

[29] J. Reardon. Improving tor using a tcp-over-dtls tunnel. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, September 2008.

[30] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), 1998.

[31] G. Shah, A. Molina, and M. Blaze. Keyboards and covert channels. In *Proceedings of the 15th USENIX Security Symposium*, July-August 2006.

[32] 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 *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, May 2002.

[33] The Tor Project, Inc. . Tor: anonymity online. `http://tor.eff.org/`, 2008.

[34] The Trustees of Princeton University. Planetlab | an open platform for developing, deploying, and accessing planetary-scale services. `http://www.planet-lab.org/`, 2008.

[35] T.Takahashi and W. Lee. An assessment of voip covert channel threats. In *Proceedings of the IEEE International Conference on Security and Privacy in Communication Networks (SecureComm)*, September 2007.

[36] X. Wang, S. Chen, and S. Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In *Proceedings of the IEEE Symposium on Security & Privacy (S&P)*, May 2007.

[37] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer voip calls on the internet. In *Proceedings of the 12th ACM Conference on Computer Communications Security (CCS)*, November 2005.

[38] X. Wang and D. S. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter-packet delays. In *Proceedings of the 2003 ACM Conference on Computer and Communications Security (CCS)*, November 2003.

[39] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao. Dsss-based flow marking technique for invisible traceback. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P)*, 2007 May.

[40] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao. On flow correlation attacks and countermeasures in mix networks. In *Proceedings of Workshop on Privacy Enhancing Technologies (PET)*, May 2004.

Appendix A

Algorithm 1 shows the signal recovery mechanism with continuously embedded bits at a malicious Tor entry node and 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;

$i = 0$; $j = 0$
while $i \leq m$ **do**
  if $C[i] == S[j]$ **then**
    Signal S[j] is matched.
  else if $C[i] < S[j]$ **then**
    Signal S[j] is splitted.
    if $C[i] + C[i+1] == S[j]$ **then**
      Signal $S[j]$ is processed as Type I with $k = 1$.
    else if $C[i] + C[i+1] > S[j]$ **then**
      Signal $S[j]$ and $S[j+1]$ are processed as Type II with $k = 1$.
    else if $C[i] + C[i+1] < S[j]$ **then**
      Find the value of k
      if $C[i] + \ldots + C[i+k] == S[j]$ **then**
        Signal $S[j]$ is processed as Type I with $k \geq 2$.
      else
        Signal $S[j]$ and $S[j+1]$ is processed as Type II with $k \geq 2$.
      end if
      $i = i + k$;
    end if
  else if $C[i] > S[j]$ **then**
    Two or more signals are combined together.
    if $C[i] == S[j] + S[j+1]$ **then**
      Signal $S[j]$ and $S[j+1]$ are processed as Type III with $k = 1$.
    else if $C[i] < S[j] + S[j+1]$ **then**
      Signal $S[j]$ and $S[j+1]$ are processed as Type IV with $k = 1$.
    else if $C[i] > S[j] + S[j+1]$ **then**
      Find the value of k
      if $C[i] == S[j] + \ldots + S[j+k]$ **then**
        These combined signals are processed as Type III with $k \geq 2$.
      else
        These combined signals are processed as Type IV with $k \geq 2$.
      end if
      $j = j + k$
    end if
  end if
  $i = i + 1$; $j = j + 1$
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 non-watermark cells.

$i = 0$; $j = 0$
while $i \leq m$ **do**
  Remove the non-watermark packets $Q[j]$ from $C[i]$.
  while $Q[j] > C[i]$ **do**
    $C[i+1] = C[i+1] + C[i]$
  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