Detecting Phantom Communication using Cell Counting Attack

S. Kavya, CH. Aruna

Abstract: Various low-latency anonymous communication systems have been designed to provide anonymity service for users. To hide the communication of users, the anonymity systems such as (TOR) pack the application data into equal – sized cells. The size of IP packets in the Tor network can be very dynamic and the IP layer may be repack cells. A new cell-counting attack against Tor allows the attacker to confirm anonymous communication relationship among users very quickly. By varying the number of cells in the target traffic at the malicious exit onion router, the attacker can embed a secrete signal into variation of cell counter of the target traffic and it will be carried and arrive at the malicious entry onion router. Then an accomplice of the attacker will detect the signal based on received cells and confirm the communication among the users. The main feature of this attack is, highly efficient and confirms very short communication session with only tens of cells.

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

I. Introduction

Fears about privacy and security have received greater attention with the speedy growth and public apperception 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. Formerly, researchers have developed numerous anonymous communication systems such as message-based (high-latency) or flow-based (low-latency) anonymity applications. E-mail is a typical example for message-based anonymity application. 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.

To degrade the anonymity service provided by anonymous communication systems, traffic analysis attacks have been studied. Existing traffic analysis attacks can be categorized into two groups: passive analysis and active watermarking techniques. Passive 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.

Manuscript received October, 2014.

***S. Kavya**, M.Tech Scholor, Department of Computer Science and Engineering, KKR & KSR Institute of Technology and Sciences, Guntur, INDIA.*

***CH. Aruna**, Asst. Professor, Department of Computer Science and Engineering, KKR&KSR Institute of Technology and Sciences, Guntur, INDIA.*

A passive packet-counting scheme to observe the number of packets of a connection that arrives at a mix node and leaves a node. Because this type of attack relies on comparing the timings of messages moving through the anonymous system and does not change the traffic characteristics, it is also called as passive *timing* attack.

To improve the accuracy of this type of attacks, the active watermarking technique has recently received much attention. The idea of this technique is to actively introduce special signals into the sender's outbound traffic with the intention of recognizing the embedded signal at the receiver's inbound traffic.

In this paper, we focus on the active watermarking technique, which has been active in the past few years. 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. This introduced the problem of formalized as detecting a known spread-spectrum signal with noise caused by network dynamics. And even 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, to the best of our knowledge, no existing work can meet all these requirements simultaneously.

In this paper, we investigate a cell-counting-based attack against 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.

II. Literature Study

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

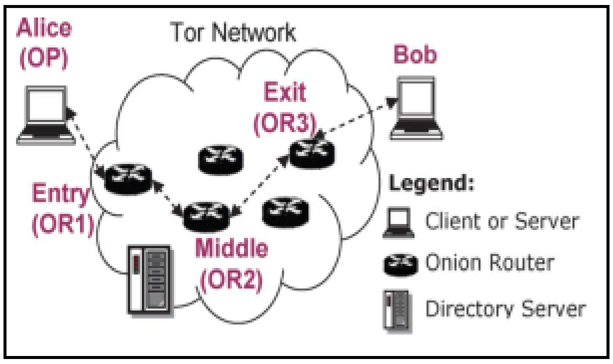

Fig. 1: Network architecture of TOR

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

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, tunnelling through this connection, *OP* sends a *CELL_CREATE* cell and uses the *Diffie–Hellman* (DH) handshake protocol to negotiate a base key $\mathbf{K_1} = \mathbf{g}^{xy}$ 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. 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.

C. Processing Cells at Onion Routers

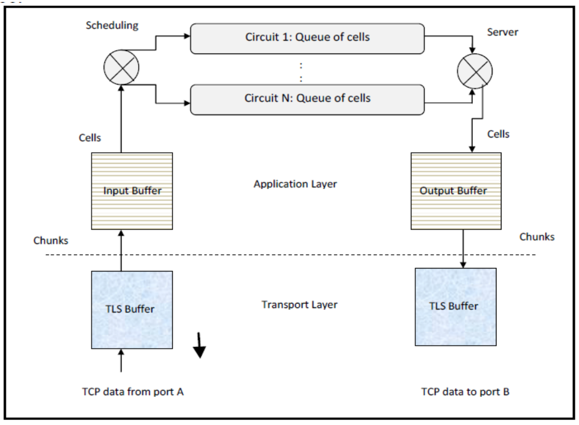

Fig. 2: Processing the cells at onion routers

Fig. 2 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 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.

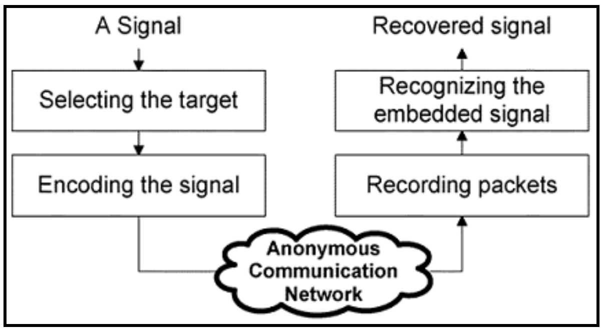

Fig. 3: Cell-counting attack

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.

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.
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 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]. 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.

As we stated, 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 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.

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. A co-conspirator 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. 3, the workflow of the cell-counting 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.

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. A co-conspirator 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, a co-conspirator 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. 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 next section.

C. Issues and Solutions

From the explanation 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 unclear 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.

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 misrepresentation of the signal, we present a recovery mechanism that robustly recognizes the embedded signal.

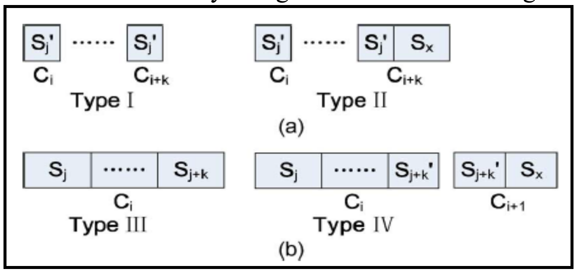

Fig. 4: Signal division and combination

The combination or division of the cells for different units of the signal can be categorized into four types. Fig. 4(a) illustrates two types of the cell division for the unit of the signal, and Fig. 4(b) illustrates the two types of the cell combination for different units of the signal. Let $C=\{C_0,C_1,C_2,...,C_i,....,C_{m-1}\}$ be the cell numbers recorded in the circuit queue at the entry onion router, and $C_i \in ([0, m-1])$ is the number of the cells, which is a positive integer. Recall the original signal is denoted as $S=\{S_0,S_1,S_2,...,S_j,....,S_{n-1}\}$. Let S_j be the j^{th} signal bit, S_j' as the part of the j^{th} signal bit, and let S_x be the integral signal bits or a remaining signal bit in the packet or a null signal bit.

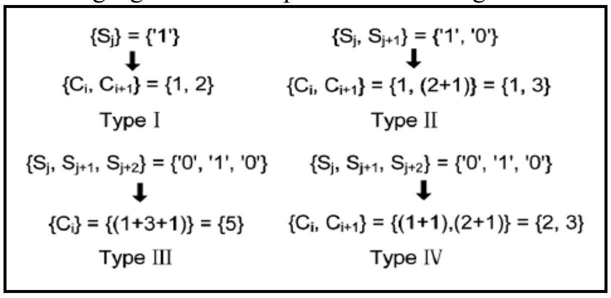
{S_j} = {'1'}
{C_i, C_{i+1}} = {1, 2}
Type I
{S_j, S_{j+1}} = {'1', '0'}
{C_i, C_{i+1}} = {1, (2+1)} = {1, 3}
Type II
{S_j, S_{j+1}, S_{j+2}} = {'0', '1', '0'}
{C_i} = {(1+3+1)} = {5}
Type III
{S_j, S_{j+1}, S_{j+2}} = {'0', '1', '0'}
{C_i, C_{i+1}} = {(1+1),(2+1)} = {2, 3}
Type IV

Fig. 5: Examples of signal division and combination

Type-I distortion indicates that the original signal S_j is divided into $k+1$ separate cells. Fig. 5 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. 5 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. 5 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 all together. Type-IV distortion indicates that a part of S_{j+k} is merged into the following cells. Fig. 5 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. 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. EXPERIMENTAL EVALUATION

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 below Fig.6, 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 **Location1**. The server (Bob), and the exit onion router is at **Location2**. All computers are on different IP address segments and connected to different Internet service providers (ISPs). Fig. 6 shows the experiment setup.

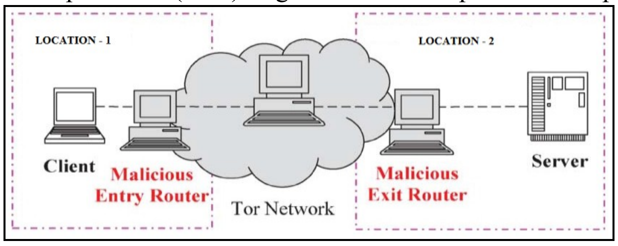

Fig. 6: Experiment setup

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. 6. 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 send a file from the client (Alice). By using the Tor configuration file and manipulatable parameters, such as *EntryNodes*, *ExitNodes*, *StrictEntryNodes*, and *StrictExitNodes*, we let the client choose both the malicious entry and exit onion routers along the circuit.

V. 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.

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, andW. 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 secondgeneration 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 connectionbased anonymity systems," in *Proc. ESORICS*,Oct. 2003, pp. 116–131.

[8] B. N. Levine,M. K. Reiter, C.Wang, andM. 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, "Lowresource 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, "Lowresource 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 cellsmean 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.