ISSN: 2321-7782 (Online)
Volume 2, Issue 1, January 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

Analysis of a new cell-counting-based attack against connection based Tor

Kiran P. Somase[1]
Student
Department of Computer Science Engineering
Siddhi Vinayak College of Science and Higher Education
Alwar (Rajsthan)
India

Neeru Yadav[2]
Asst. Prof & HOD
Department of Computer Science Engineering
Siddhi Vinayak College of Science and Higher Education
Alwar (Rajsthan)
India

Abstract: ***The onion router allows hiding your identity. Various software are under this categories are available that allows online anonymity. It doesn't allow network surveillance or traffic analysis to get tracked. But most these software used equal sized cells (512B).In this paper we are analysing cells counting attacks against Tor which allows identify anonymous communication among user. Firstly we need to buy one onion router then attacker can include special signal l i.e. cell counting attack into network traffic. We have developed this system against Tor and experimental results show the effectiveness of algorithms and allow detecting Tor network.***

Keywords: Anonymity, cell counting attack, anonymous network, Tor, Malicious Router

I. INTRODUCTION

In this project, we will focus on the active watermarking technique, which has been active in the past few years. Proposed a flow-marking scheme based on the direct sequence spread spectrum technique by utilizing a pseudo-noise 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. proposed a non-blind network flow watermarking scheme called RAINBOW for stepping stone detection.

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

In this paper, we investigate a new cell-counting-based attack against Tor, a real-world, circuit-based low-latency anonymous communication network. This attack is a novel variation of the standard timing attack. It can confirm anonymous communication relationship among users accurately and quickly and is difficult to detect. In this attack, the attacker at the malicious exit router detects the data transmitted to a suspicious destination (e.g., server Bob). The attacker then determines whether the data is a relay cell or a control cell in Tor. After excluding the control cells, the attacker manipulates the number of relay cells in the circuit queue and flushes out all cells in the circuit queue. In this way, the attacker can embed a signal (a series

of "1" or "0" bits) into the variation of the cell count during a short period in the target traffic. An accomplice of the attacker at the entry onion router detects and excludes the control cells, records the number of relay cells in the circuit queue, and recovers the embedded signal. The signal embedded in the target traffic might be distorted because the cells carrying the different bits (units) of the original signal might be combined or separated at middle onion routers.

To address this problem, we have developed the recovery algorithms to accurately recognize the embedded signal. our theoretical analysis shows that the detection rate is a monotonously increasing function with respect to the delay interval and is a monotonously decreasing function of the variance of one way transmission delay along a circuit. In our real-world experiments, the experimental results match the theoretical results well. To be specific, our attack needs only 2 s to achieve a true positive rate of almost 100% and the false positive rate of almost 0%. We have implemented the cell-counting-based attack against Tor and performed a set of real-world Internet experiments to Fig. 1. Tor network 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.

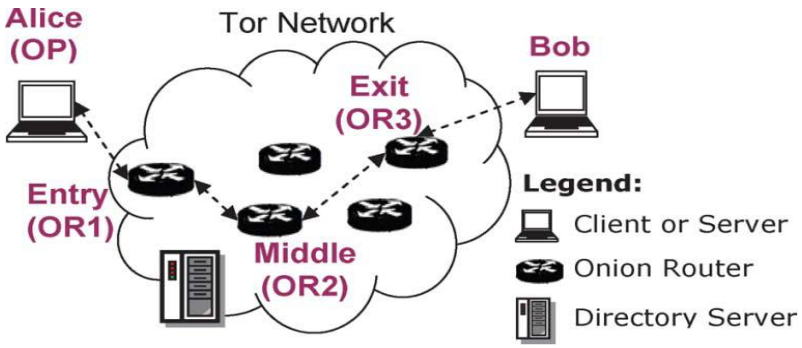

Fig 1. Tor network

Second, this attack is effective, and its detection rate approaches 100% with very low false positive rate. Third, the short and secret signal makes it difficult for others to detect the presence of the embedded signal. Our time-hopping-based signal embedding technique makes the attack even harder to detect. The attack poses a significant threat to the anonymity provided by Tor because the attack can confirm over half of communication sessions by injecting around 10% malicious onion routes on Tor [18], [19].

II. BACKGROUND

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

A. Components of Tor

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

1) *Alice* (i.e., *Client*): The client runs 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. Fig. 2 illustrates the cell format used by Tor. All cells have a 3-B header, which is not encrypted in the onion-like fashion so that the intermediate Tor routers can see this header. The other 509 B are encrypted in the onion-like fashion. There are two types of cells: *control* cell shown in Fig. 2(a) and *relay* cell shown in Fig. 2(b). The command field (*Command*) of a control cell can be: *CELL_PADDING*, used for keepalive and optionally usable for link padding, although not used currently; *CELL_CREATE* or *CELL_CREATED*, used for setting up a new circuit; and *CELL_DESTROY*, used for releasing a circuit. The command field (*Command*) of a relay cell is *CELL_RELAY*. Note that relay cells are used to carry TCP stream data from Alice to Bob. The relay cell has an additional header, namely the relay header.

There are numerous types of relay commands (*Relay Command*), including *RELAY_COMMAND_BEGIN*, *RELAY_COMMAND_DATA*, *RELAY_COMMAND_END*, *RELAY_COMMAND_SENDME*, *RELAY_COMMAND_EXTEND*, *RELAY_COMMAND_DROP*, and *RELAY_COMMAND_RESOLVE*. Note that all these can be found in *or.h* in released source code package by Tor.c

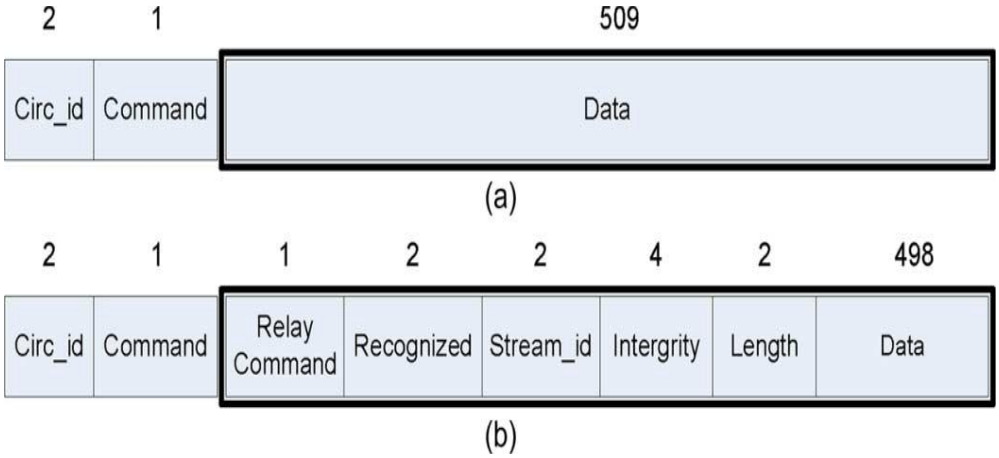

Fig. 2. Cell format by Tor. (a) Tor cell format. (b) Tor relay cell format.

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. I will now illustrate the procedure that the *OP* establishes a circuit and downloads a file from the server. *OP* first sets up a TLS connection with *OR1* using the TLS protocol. Then, tunneling through this connection, *OP* sends a *CELL_CREATE* cell and uses the *Diffie–Hellman* (DH) handshake protocol to negotiate a base key $K1=g^{xy}$ with *OR1*, which responds with a *CELL_CREATED* cell. From this base key material, a forward symmetric key kf1 and a backward symmetric key kb1 are produced [21]. In this way, a 1-hop circuit *C1* is created. Similarly, *OP* extends the circuit to a 2-hop circuit and 3-hop circuit. After the circuit is set up between the *OP* and *OR3*, *OP* sends a *RELAY_COMMAND_BEGIN* cell to the exit onion router, and the cell is encrypted as $\{\{\{\text{Begin} < \text{IP, Port} >\}_{kf3}\}_{kf2}\}_{kf1}$, 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. 3 illustrates the procedure of processing cells at onion routers. Note that the cells mentioned below are all *CELL_RELAY_DATA* cells, which are used to carry end-to-end stream data between Alice and Bob. To begin with, the onion router receives the TCP data from the connection on the given port *A*. After the data is processed by TCP and TLS protocols, the data will be delivered into the TLS buffer of the connection. When there is pending data in the TLS buffer, the read event of this connection will be called to read and process the data. The connection read event will pull the data from the TLS buffer into the connection input buffer. Each connection input buffer is implemented as a linked list with small chunks. The data is fetched from the head of the list and added to the tail. After the data in the TLS buffer is pulled into the connection input buffer, the connection read event will process the cells from the connection input buffer one by one. As stated earlier, the cell size is 512 B. Thus, 512-B data will be pulled out from the input buffer every time until the data remaining in the connection input buffer is smaller than 512 B.

Since each onion router has a routing table that maintains the map from source connection and circuit ID to destination connection and circuit ID, the read event can determine that the transmission direction of the cell is either in the forward or backward direction. Then, the corresponding symmetric key is used to decrypt/encrypt the payload of the cell, replace the present circuit ID with the destination circuit ID, and append the cell to the destination circuit queue.

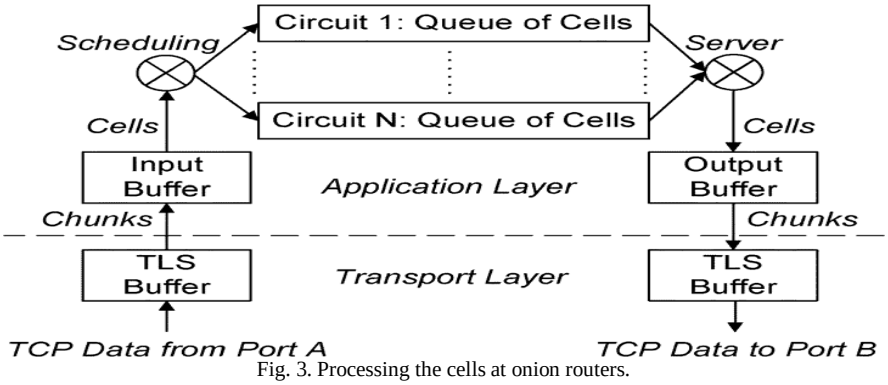

Fig. 3. Processing the cells at onion routers.

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.

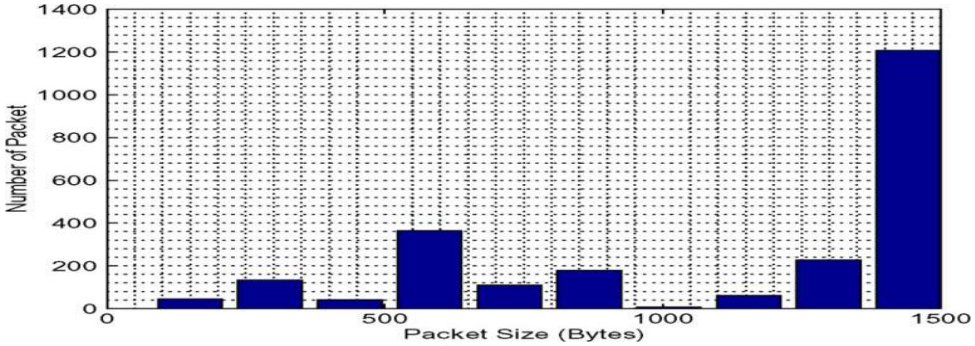

Fig. 4. Number of packets versus packet size.

III. CELL-COUNTING-BASED ATTACK

As the size of IP packets is dynamic so based on some constraints we need to initiate attack on streams.

1. Dynamic IP Packet Size of Traffic over Tor

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

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

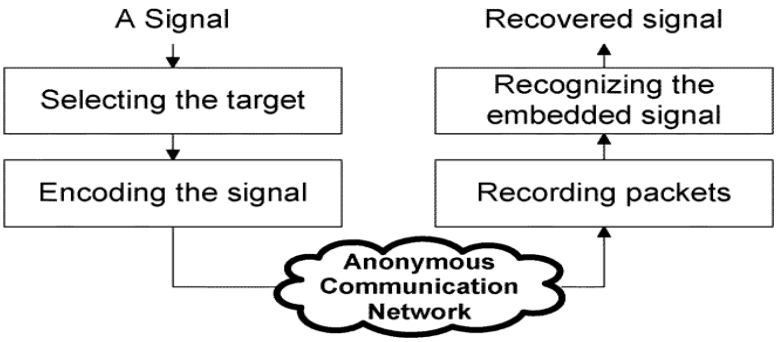

Fig. 5. Cell-counting-based attack.

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

2. Basic Idea of Cell-Counting-Based Attack

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

In order to do so, we assume that the attacker controls a small percentage of exit and entry onion routers by donating computers to Tor. This assumption is also used in other studies [3], [10], [18], [19]. The assumption is valid since Tor is operated in a voluntary manner [21]. For example, attackers may purchase Amazon EC2 virtual machines, which can be put into Tor. The attack can be initiated at either the malicious entry onion router or exit onion router, up to the interest of the attacker. In the rest of the paper, we assume that the attack is initiated at an exit onion router connected to server Bob and intends to confirm that Alice communicates with a known server Bob. The basic idea is as follows. An attacker at the exit onion router first selects the target traffic flow between Alice and Bob. The attacker then selects a random signal (e.g., a sequence of binary bits), chooses an appropriate time, and changes the cell count of target traffic based on the selected random signal. In this way, the attacker is able to embed a signal into the target traffic from Bob. The signal will be carried along with the target traffic to the entry onion router connecting to Alice. An accomplice of the attacker at the entry onion router will record the variation of the received cells and recognize the embedded signal. If the same pattern of the signal is recognized, the attacker confirms the

communication relationship between Alice and Bob. As shown in Fig. 6, the workflow of the cell-counting-based attack is illustrated as follows.

Step 1: Selecting the Target: At a malicious exit onion router connected to the server Bob, the attacker will log the information, including server Bob's host IP address and port used for a given circuit, as well as the circuit ID. The attacker uses *CELL_RELAY_DATA* cells since those cells transmit the data stream. According to the description of Tor in Section II, we know that the attacker is able to obtain the first cell backward to the client, which is a *CELL_CREATED* cell and is used to negotiate a symmetric key with the middle onion router. The second cell backward to the client will be a *CELL_RELAY_CONNECTED* cell. All sequential cells will be *CELL_RELAY_DATA* cell, and the attacker starts the encoding process shown in Step 2.

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

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

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

3. Issues and Solutions

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

1) Encoding Signals at Exit Onion Routers: Two Cells for

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

2) Proper Delay Interval Should Be Selected for Transmitting Cells: Since the signal modulates the number of cells transmitted from the exit onion router to the entry onion router, the delay intervals among cells that carry different units (bits) of the signal will have impact on the accuracy and detectability of the attack. Hence, care must be taken to select a proper interval for transmitting those cells. If the delay interval among cells is too large, users may not be able to tolerate the slow traffic rate and will choose another circuit to transmit the data. When this happens, the attack will fail. When the delay interval among cells is too small, it will increase the chance that cells may be combined at middle onion routers. Let us use one simple example to clarify this. We assume that the delay intervals for three bits "0," "1," and "0" of the signal are very small. The first cell for carrying the first bit "0" arrives at the middle onion router and is written into the queue. This first cell will be flushed into the output buffer if the output buffer is empty. The write event is added to the event queue, and the cell waits to be written to the network by the write event. Since the interval is small, the three cells for the second bit "1" and the cell for the third bit "0" also arrive at the middle onion router and stay in the circuit queue. When the write event is called, the first cell for carrying the first bit "0" will be written to the network, while the following three cells for carrying the second bit of the signal and one cell for carrying the third bit of the signal will be written to the output buffer all together. When this happens, the original signal will be distorted (i.e., the third bit "0" of the signal will be lost). Therefore, the attacker needs to choose the proper delay interval for transmitting cells. In addition, we will discuss the types of the division and combination of the cells with details in Section III-C.2.

We now check conditions that preserve units of the signal during transmission. Let S= $\{S_0, S1, S_{n-1}\}$ be the signal, a series of bits, where is the signal length and S_j (j $\in$ [0, n-1]) is 0 or 1. When S_j=1the attacker will choose three cells to encode bit "1." When S_j=0 the attacker will choose only one cell to encode bit "0." Let the time sequence of the signal that arrives at the *OR2* be $T=\{T_0,T_1,\ldots,T_{n-1}\}$ and T_{read} be the average time of calling the read event, which pulls the data of cells for each unit of the signal from the TLS buffer and write them to the circuit queue. Let T_{write} be the average time of calling the write event, which writes the cells in the output buffer to the network and flushes the cells in the circuit queue to the output buffer. Let the delay interval between two sequential bits of the signal be, and let the delay of transmitting data between *OR3* and *OR2* be . The relationship between T_i and T_{i+1} can be represented as follows:

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

Let the time of the cells for the signal arriving at the circuit queue be T^{queue}, where $T_i^{queue} = T_i + T_{read}$ Let the time of the cells for the signal arriving at the output buffer be $T^{outbuffer}$, where $T_i^{outbuffer} = T_i + T_{read} + T_{write}$ In order to avoid the combination of cells that belong to different units of a signal in the circuit queue, the cells for carrying one bit should be flushed to the output buffer or the network before the cells for carrying the next unit of the signal arrives at the circuit queue.

Let us now discuss the how arrived signal is divided and combined is get categories into four types.Let $C=\{C_0, C_1, C_2,..C_i \ldots C_{m-1}\}$ be the cell recorded at onion router entry,C_i i $\in$ [0, m-1]) no. of cells(I is positive integer). Original signal $S=\{S_0, S1, S_{n-1}\}$. Let S_j be the j^{th} signal bit, $S_j^{'}$ as a part of j^{th} signal bit and let S_x be integral signal bits.

Type I: Indicates original signal S_j is divided into k+1 separate cells with k=1. Suppose signal S_j is bit "1" number of cells should be 3. As onion router records C_i is 1and C_{i+1} is 2 will create 3 cells.

Type II: Indicates last part of S_j is merged with S_n with k=1.

Type III: Indicates k original signals are merged into signal packet.

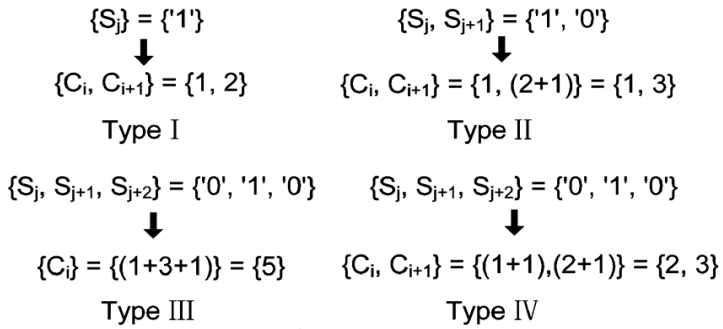

Fig 6. Examples of signal division and combination.

To deal with three types we propose algorithm 1for recovery mechanism.

Algorithm 1: Recovery Mechanism for Continuously Embedded Bits

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

```
1:  i=0, j=0;
2:  while i<=m do
3:        if C[i] ==S[j] then
4:              Signal S[j] is matched.
5:              else if C[i] < S[j] then
6:                    Signal S[j] is splitted.
7:              if C[i] +C [i+1] ==S[j] then
8:                    Signal S[j] is processed as Type I with k=1.
9:              else if C[i] + C[i+1]>S[j]then
10:                   Signal S[j] and S[j+1] are processed as Type II with k=1.
11:             else if C[i] + C[i+1]<S[j] then
12:                   Find the value of k
13:                   if C[i] +….+C[i+k]==S[j] then
14:                   Signal S[j] is processed as Type I with k>=2.
15:                   else
```



```
16:                         Signal S[j] and S[j] is processed as Type II with k>=2 .
17:                 end if
18:                  i=i+k;
19:           end if
20:     else if C[i]>S[j] then
21:           Two or more signals are combined together.
22:           if C[i]==S[j]+S[j+1] then
23:                 Signal S[j] and S[j+1] are processed as Type II with k=1 .
24:           else if C[i] <  S[j]+S[j+1] then
25:                  Signal S[j] and S[j+1] are processed as Type IV with k=1.
26:           else if C[i] > S[j]+S[j+1] then
27:                 Find the value of k.
28:                 if C[i] > S[j]+…+S[j+k] then
29:                       These combined signals are processed as Type III with
                          k>=2.
30:                 else
31:                       These combined signals are processed as Type IV with
                          k>=2.
32:                 end if
33:                 j=j+k;
34:           end if
35:     end if
36:     i=i+1,j=j+1;
37: end while
```

Once original signals are identified entry point onion router carry information of server IP with port TCP stream.

IV. EXPERIMENTAL EVALUATION

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

A. Experiment Setup

In my experiment setting illustrated in Fig. 7, we deployed two malicious onion routers as the Tor entry onion router and exit onion router. The entry onion router and client (Alice) located in Asia are deployed on PlanetLab [36]. The server (Bob) is located at one university campus in North America, and the exit onion router is at an off-campus location in North America as well. All computers are on different IP address segments and connected to different Internet service providers (ISPs). Fig. 7 shows the experiment setup. I have 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. 7. The middle onion router is selected using the default routing selection algorithm released by Tor. As I stated earlier, the cell-counting-based attack intends to confirm whether the client (Alice) communicates with the server (Bob). For verification purpose, we set up a server (Bob) and download a file from the client (Alice).

The downloading software at the client is the command line utility *wget*. By configuring *wget*'s parameters of *http_proxy* and *ftp_proxy*, I let *wget* download files through *Privoxy*, the proxy server used by Tor. By using the Tor configuration file and

manipulatable parameters, such as *EntryNodes*, *ExitNodes*, *StrictEntryNodes*, and *StrictExitNodes* [23], we let the client choose both the malicious entry and exit onion routers along the circuit.

B. Experimental Results

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

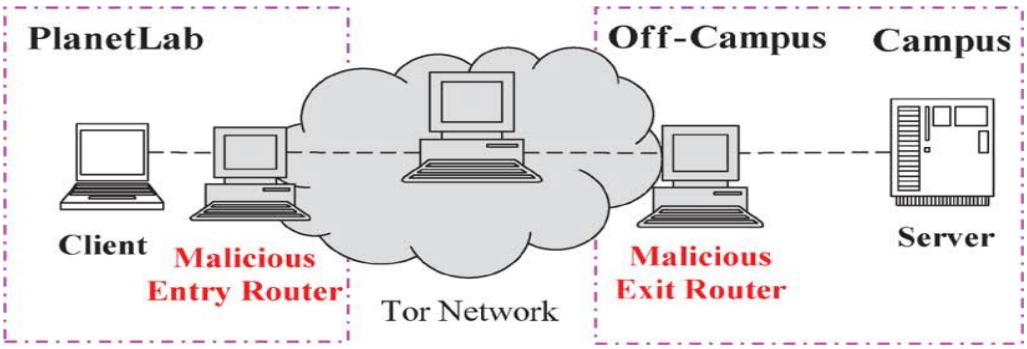

Fig. 7. Experiment setup.

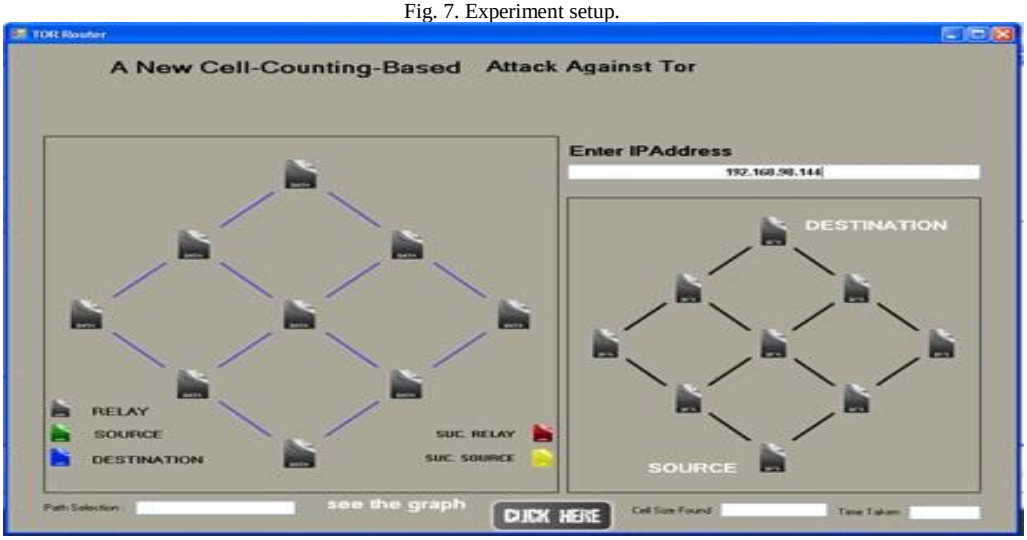

Fig. 8: Packet route simulation

Above figure shows that at destination side the packet is routed to various anonymous routers, then get received at exit anonymous router and finally received by the client.

V. Experimental Results

We developed this system using C# i.e. .net. Firstly we created fully created and transferred files over network. We transfer 35 files is of size 10MB.We have embedded signal for short period of time and recovered signals traffic marked by our signal then detection of relationship rate is approaches to 90% by the proposed method.

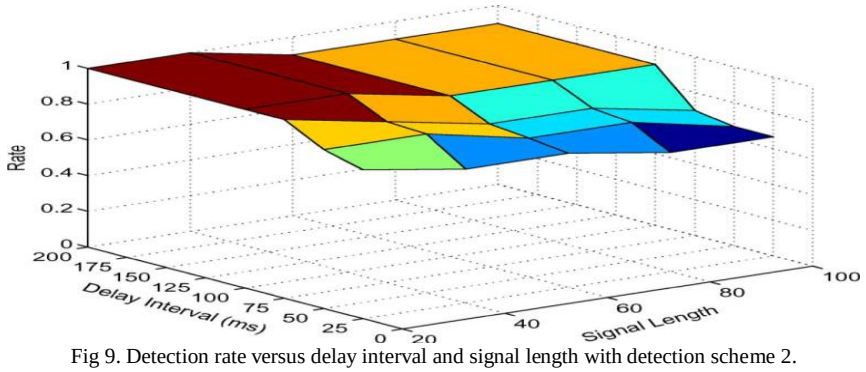

Fig 9. Detection rate versus delay interval and signal length with detection scheme 2.

VI. CONCLUSION

In this paper, we have introduced a novel cell-counting-based attack against Tor. This attack is difficult to detect and quickly confirm the anonymous communication relationship among client and server. An attacker with onion router with slightly modify target stream TCP signal. Recovery algorithm is used to recover bits at entry level onion router which shows anonymity server like Tor. As a part of Tor defending this attack will be complex and challenging task. We will keep this for future research.

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

Algorithm 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.

```
1: i=0;j=0;
2: while i<=m do
3:    Remove the non-watermark packets Q[j] from C[i].
4:    while Q[j] >C[i]do
5:              C[i+1]=C[i+1]+C[i]
6:    end while
7:   if Q[j]==C[i] then
8:              i=i+1; Q[j] is removed.
9:              Detect S[j] with C[i] by using Algorithm 1
10:  else if Q[j] < C[i] then
11:             The signal S[j] is combined with Q[j].
12:             C[i]=C[i] -Q[j];
13:             Detect S[j] with C[i] by using Algorithm 1
14:  end if
15:  i=i+1; j=j+1;
16: end while
```

References

1. Zhen Ling, Junzhou Luo, Wei yu,"A New cell-counting based Attack against Tor ," in Proc. IEEE/ACM Transactions on networking.
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.1120.

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," inProc. 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.114.
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.
40. M. Liberatore and B. N. Levine, "Inferring the source of encrypted HTTP connections," in *Proc. ACM CCS*, Oct. 2006, pp. 255–263.
41. C. V.Wright, L. Ballard, F. Monrose, and G.M.Masson, "Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob?," in *Proc. 16th Annu. USENIX Security Symp.*, Aug. 2007, pp. 43–54.
42. C. V.Wright, L. Ballard, S. E. Coull, F.Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversation," in *Proc. IEEE S&P*, May 2008, pp. 35–49.
43. X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter-packet delays," in *Proc. ACM CCS*, Nov. 2003, pp. 20–29.
44. P. Peng, P. Ning, and D. S. Reeves, "On the secrecy of timing-based active watermarking trace-back techniques," in *Proc. IEEE S&P*, May 2006, pp. 335–349.
45. Y. J. Pyun, Y. H. Park, X. Wang, D. S. Reeves, and P. Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *Proc. IEEE INFOCOM*, May 2007, pp. 634–642.

AUTHOR(S) PROFILE

Mr. Kiran P. Somase received the Diploma in Computer Technology from K.B.P. Polytechnic, Kopargaon in 2006 and B.E. degree in Computer Engineering from SRES College of Engineering, Kopargaon in 2009. Also currently pursuing the M.Tech in Computer Science Engineering from Siddhi Vinayak College of science and higher Education, Alwar affiliated to Rajasthan Technical University, Kota.

Ms. Neeru Yadav completed M.Tech in Computer Science from Banasthali University in 2008 and B.Tech in Computer Science from Rajasthan University in 2006. Presently working as Assist. Prof. in Siddhi Vinayak College of Sci. & Hr. Education, Alwar, Rajasthan Since 2008.