PREVENTION AGAINST NEW CELL COUNTING ATTACK AGAINST TOR NETWORK

Gurudas V R

Student, Department of Computer Science and Engineering, Jawaharlal Nehru National College of Engineering, Shimoga.VTU, Karnataka, India, ***gurudasvr@gmail.com***

Abstract

Various low-latency anonymous communication systems such as Tor and Anonymizer have been designed to provide anonymity service for users. In order to hide the communication of users, most of the anonymity systems pack the application data into equal-sized cells (e.g., 512 B for Tor, a known real-world, circuit-based, low-latency anonymous communication network). Via extensive experiments on Tor, authors in [22] found that the size of IP packets in the Tor network can be very dynamic because a cell is an application concept and the IP layer may repack cells. Based on this finding, they investigated a new cell-counting-based attack against Tor, which allows the attacker to confirm anonymous communication relationship among users very quickly. In this attack, by marginally varying the number of cells in the target traffic at the malicious exit onion router, the attacker can embed a secret signal into the variation of cell counter of the target traffic. The embedded signal will be carried along with the target traffic and arrive at the malicious entry onion router. Then, an accomplice of the attacker at the malicious entry onion router will detect the embedded signal based on the received cells and confirm the communication relationship among users. In this paper i propose a mechanism to prevent from this attack.

Keywords: *security; sensor networks, denial of sleep attack*

----------***----------

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 (LBSs), and E-voting. In these applications, encryption alone cannot maintain the anonymity required by participants [1]–[3]. In the past, researchers have developed numerous anonymous communication systems. Generally speaking, mix techniques can be used for either message-based (high-latency) or flow-based (low-latency) anonymity applications. E-mail is a typical message-based anonymity application, which has been thoroughly investigated [4]. Research on flow-based anonymity applications has recently received great attention in order to preserve anonymity in low-latency applications, including Web browsing and peer-to-peer file sharing [5], [6].

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

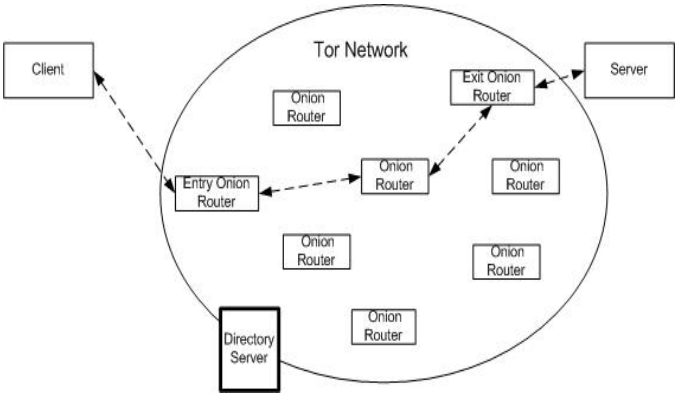


Figure 1: Tor Network.

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

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

A successful attack against anonymous communication systems relies on accuracy, efficiency, and detectability of active watermarking techniques. Detectability refers to the difficulty of detecting the embedded signal by anyone other than the attackers. Efficiency refers to the quickness of confirming anonymous communication relationships among users.

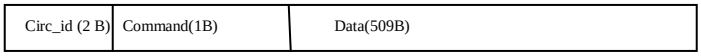
Authors in [22] proposed a new cell-counting-based attack against Tor, a real-world, circuit-based low-latency anonymous communication network. This attack is a novel variation of the standard timing attack. It can confirm anonymous communication relationship among users accurately and quickly and is 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, authors developed the recovery algorithms to accurately recognize the embedded signal.

In this paper, we study the above problem in detail and propose a solution to prevent this kind of attack in the network.

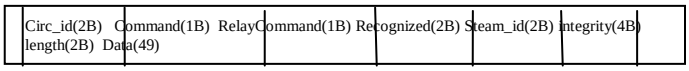
2. LITERATURE SURVEY

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

Tor is a popular overlay network for providing anonymous communication over the Internet. It is an open-source project and provides anonymity service for TCP applications [20]. As shown in Fig. 1, there are four basic components in Tor.
1) Alice (i.e., Client): The client runs a local software called onion proxy (OP) to anonymize the client data into Tor.
2) Bob (i.e., Server): It runs TCP applications such as a Web service.
3) Onion routers (ORs): Onion routers are special proxies that relay the application data between Alice and Bob. In Tor, transport-layer security (TLS) connections are used for the overlay link encryption between two onion routers. The application data is packed into equal-sized cells (512 B as shown in Fig. 2) carried through TLS connections.
4) Directory servers: They hold onion router information such as public keys for onion routers. Directory authorities hold authoritative information on onion routers, and directory caches download directory information of onion routers from authorities. A list of directory authorities is hard-coded into the Tor source code for a client to download the information of onion routers and build circuits through the Tor network.



2(a)



2(b)

Figure 2: Cell Format.

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 keep alive 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. In Tor, an OR maintains a TLS connection to other ORs or OPs on demand. The OP uses a way of source routing and chooses several ORs (preferably ones with high bandwidth and high uptime) from the locally cached directory, downloaded from the directory caches. The number of the selected ORs is referred as the path length. Here there is 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.

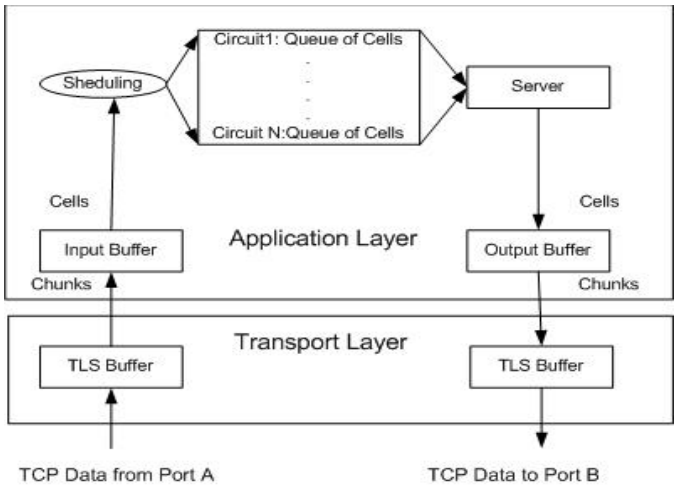


Figure 3: Processing of cells.

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

Then, the write event will pull as many cells as possible from the circuit queue of the currently active circuit to the output 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 COUNTING ATTACK

As i 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. 4, the workflow of the cell-counting-based attack is illustrated as follows.

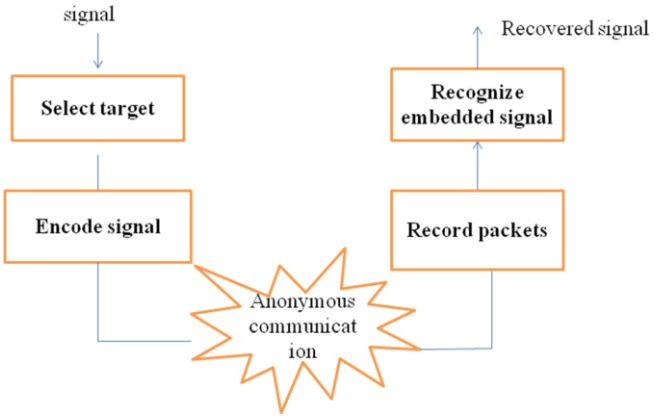


Figure 4: Workflow of Attack.

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 previous section, there was introduction of 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. 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 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.

4. DETAILS OF PROPOSED ATTACK PREVENTION METHOD

As we discussed the attack in the previous section, we found that by variation of cell relay packet processing at exit onion router, the attacker is able to encode certain signals for each flow and thereby was able to break the anonymity of the network. To eliminate this variation, i propose a solution based on the onion routers in the mid routers in the TOR networks.
The basic of the proposed solution is that onion routers at the middle layer has to observe the number of RELAY packets being forwarded for a connection id in the upward direction

and must maintain the same value for the RELAY packets arriving from the onion router. This way it ensures that for the connection arriving from a entry router, there is no variation in the relay packet forwarding at all. By denying the variations the attacker at the exit router even though inserts variation by flushing the relay packet differently, at the middle router uses the same fixed rate, this makes no variation appear at the entry router.

The detailed working of the proposed solution is given below
At the mid onion routers following processing occur. A mid onion router may be catering to packet from different incoming onion router. For each of them it has to allocate unique outgoing flow rate.

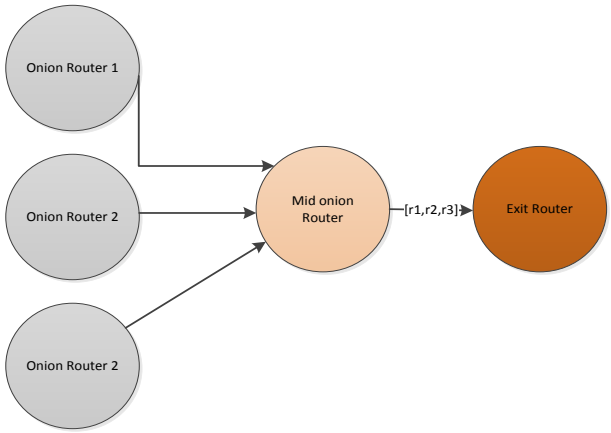


Figure 5: processing in Mid Router.

For the flow from different routers, the mid onion router has to allocate a different outgoing rate of relay packets and it must maintain the same value for all the connection from that onion router.

If this procure is followed in all the mid routers, any entry onion router for all the incoming connections, the same rate of relay packet is maintained. This makes the entry router not able to detect any signal.

At the mid onion router, to ensure the rate packet must be buffered. This will consume extra memory space at the mid onion routers but to provide anonymity this cost is bearable.

5. PERFORMANCE ANALYSIS

I implement the proposed solution in NS2 and analyze the performance of the system in terms of average buffer needed at each mid onion router. I do this for TOR network of size of 20 and i create many TOR connections from 20 to 40 connections per seconds between the nodes and measure the average increase in buffer size at each router.

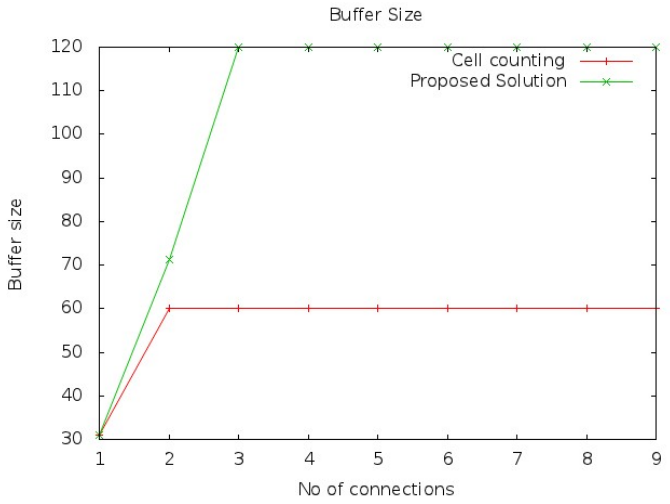


Figure 6: Number of Connections per second versus buffer size

Through performance i check the average buffer size needed for two connection durations of 1min and 2 min and see that buffer size needed. This additional buffer size must be provisioned at the router for providing safety against anonymity leakage attacks.

CONCLUSIONS

In this paper, i introduced a prevention solution against cell-counting-based attack against Tor. By disallowing the variation for shorter period we are able to stop attackers from inserting signal by using the variation in processing of relay cell. By denying this variation, we are able to avoid the cell counting attack and prevent the anonymity in the tor network.

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," 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].
[20] Available: http://www.blackhat.com/presentations/bh-dc-09/Fu/ BlackHat-DC-09-Fu-Break-Tors-Anonymity.pdf
[21] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: Anonymity online," 2008 [Online]. Available: http://tor.eff.org/index.html.en.
[22] Zheng Ling "A New cell counting based attack against TOR".