

Equal-sized Cells Mean Equal-sized Packets in Tor?

Zhen Ling*, Junzhou Luo*, Wei Yu[†] and Xinwen Fu[‡]

*Southeast University, Nanjing 211189, P. R. China. {zhenling, jluo}@seu.edu.cn

[†]Towson University, Towson, MD 21252, USA. wyu@towson.edu

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

Abstract—Tor is a well-known low-latency anonymous communication system. To prevent the traffic analysis attack, Tor packs application data into equal-sized cells. However, we found that equal-sized cells at the *application layer* do not necessarily produce equal-sized packets at the *network layer*. Therefore, we introduced a packet size based attack that compromises Tor’s communication anonymity with no need of controlling Tor routers. An attacker can manipulate size of packets between a web site and an exit onion router and embeds a signal into the target traffic. An accomplice at the user side can sniff the traffic and recognize this signal. To cope with the signal distortion incurred by Tor and Internet, we developed an effective signal recovery mechanism. Our real-world experiments validate the effectiveness of our attack against Tor. Our work demonstrates the need for re-considering the issue of padding anonymous communication data into equal size.

Index Terms—Packet, Cell, Anonymity, Tor

I. INTRODUCTION

Tor [1] is a popular anonymous communication system. In this paper, we present a packet size based attack that may drastically degrade Tor’s communication privacy. This attack manipulates traffic into Tor and belongs to the active traffic analysis attack [2], [3], [4], [5]. The idea of this attack is that an attacker actively embeds a secret signal into target traffic entering the Tor network and its accomplice is able to recognize the signal in the traffic leaving Tor to the victim client. This attack can reduce the false positive rate significantly and don’t require massive traffic training [6], [7]. In addition, this attack requires much less resource than the approach in [5] because the latter requires the control of a necessary set of Tor routers.

There has been extensive research work done on attacks degrading anonymous communication through Tor. Most existing approaches are based on traffic analysis [8], [6], [7], [9], [10], [2], [3], [4]. These 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 correlation between sender’s outbound traffic and receiver’s inbound traffic based on statistical measures [7], [6]. This type of technique requires a relatively long period of traffic observation for a reasonable detection rate. Active watermarking technique can reduce attack lasting time, improve attack success rate and has recently received more attention. The idea 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 [11], [3], [5].

In this paper, we focus on the active watermarking technique. In our newly proposed attack, the attacker between a web site and the victim client accessing the web site via Tor

can embed a secret signal into the packet size variation of target traffic. This attacker can be the owner of a malicious web server or one manipulating (repacketizing) the traffic between the web server and the onion router. Without loss of generality, we use the former case as an example in this paper. To encode bit 1 or bit 0 of a signal, the attacker at the web site manipulates the size of application data sent at one time to a Tor router, e.g., a large chunk of application data refers to bit 1 and a small chunk refers to bit 0. Although Tor routers on a circuit pack the incoming data into equal-sized cells at the application layer, these cells are further packed into IP packets when they are transmitted at the network layer and to the victim client. The attacker’s accomplice at the victim side will be able to recognize the signal since the large chunk of application data as bit 1 may correspond to large IP packet size and small chunk may incur small IP packet size. To cope with packet size distortion caused by Tor and Internet (e.g., packet padding, packet merging, limited TCP buffer and various MTU), we design a detection mechanism to recover the signal. In this way, the anonymity service provided by Tor is degraded to some extent.

We implemented this packet size based attack against Tor and performed the real-world experiments over Tor. The experimental results demonstrate the feasibility and effectiveness of our investigated attack, which is simple, efficient and effective. Our data show that only *tens of packets* are needed for our packet size based attack to achieve a high detection rate. At the same time, the false positive is very low. In comparison with related attacks [5] and [2], our investigated attack needs less resource (e.g., without controlling any Tor router) and requires just tens of packets to achieve a high detection rate and low false positive rate. Our results clearly demonstrate that equal sized cells at the application layer cannot thwart an attacker who exploits the IP packet size at the network layer to degrade the anonymity service provided by Tor.

The remainder of this paper is organized as follows: We introduce Tor network components and review the existing cell counting based attack against Tor in Section II. In Section III, we present the basic idea of the packet size based attack. We also address several issues about the attack and propose the solutions. Extensive experimental results are presented in Section IV. We review related work in Section V and conclude this paper in Section VI.

II. BACKGROUND

In this section, we first present Tor network components and then review the existing cell counting based attack against Tor [5].

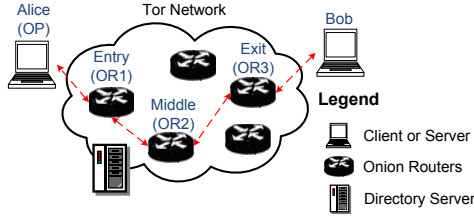


Fig. 1. Tor Network

A. Components of Tor Network

Tor is a real-world, circuit-based low-latency anonymous communication network. It is an open source project and provides anonymity service for TCP applications [12]. Figure 1 illustrates the basic components of Tor network [13]. As shown in Figure 1, it has the following four basic components:

- *Alice* (i.e. *Client*) runs a local software called *onion proxy* (*OP*) to anonymize the client data into Tor.
- *Bob* (i.e. *Server*) runs TCP applications such as a web service and anonymously communicates with *Alice* over Tor.
- *Onion routers* (*OR*) are special proxies that relay the application data between Alice and Bob. By default, *client* uses three ORs to relay their data. Denote these three ORs as entry router (*OR1*), middle router (*OR2*) and exit router (*OR3*), respectively. In Tor, Transport Layer Security (TLS) connections are used for the overlay link encryption between two onion routers. The application data is packed into equal-sized cells (512 bytes as shown in Figure 2) transmitted through TLS connections.
- *Directory servers* hold onion router information such as public keys for onion routers. Directory authorities hold authoritative information on onion routers and directory caches the downloaded directory information of onion routers from authorities.

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

B. Cell Counting Based Attack against Tor

The most related work to this proposed attack is the cell counting based attack against Tor proposed by Ling *et al.* [5]. However, this attack assumes that both *OR3* and *OR1* over the user’s circuit are controlled by the attacker. To this end, this attack takes advantage of the cell processing in the queue as shown Figure 3. When the user downloads a file from a web site, the attacker at the *OR3* side manipulates the cells at the application layer over Tor to embed the signal. To encode the “0” bit of original signal, an adversary calls the read event and pulls one cell from the input buffer to the queue, and then calls the write event to flush this cell into output buffer promptly. Then the cell can be written into the TLS connection in the next coming write event. Likewise, in order to encode the “1” bit of original signal, the attacker flushes three cells into the TLS connection. In this way, the signal is embedded into the user’s traffic. An accomplice at the *OR1* side counts the number of the incoming cells to recognize the embedded signal. Once the sequence of the signal is detected,

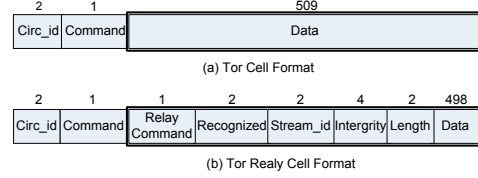


Fig. 2. Cell Format By Tor

the communication relationship between the users and web site is confirmed.

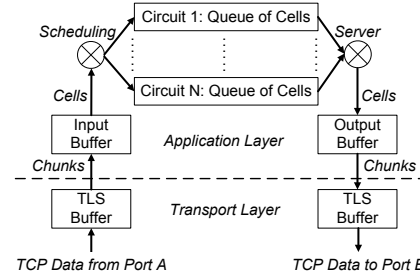


Fig. 3. Processing the Cells at Onion Routers [5]

As we can see, the attack proposed in [5] has to control both *OR3* and *OR1*. It would require very high bandwidth and up-time for the onion router to become *OR1*, i.e., entry onion router. Differently, the attack proposed in this paper does not have such constraint and our real-world experimental results over Tor clearly demonstrate that equal sized cells at the application layer cannot thwart an attacker who may exploit the IP packet size at the network layer to degrade the anonymity service provided by Tor.

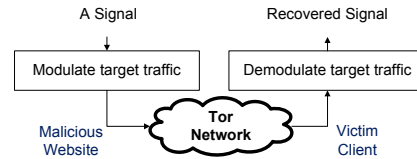


Fig. 4. Workflow of Packet Size based Attack

III. PACKET SIZE BASED ATTACK AGAINST TOR

In this section, we first introduce the basic idea of our packet size based attack against Tor. We then present some challenging issues and corresponding solutions, followed by discussion.

A. Basic Idea

Without loss of generality, we assume that an attacker owns a malicious web site and intends to find out who accesses the web sit. Note that the attacker does not need to control any onion router in the Tor network. The basic idea of this attack is as follows: the attacker first controls a reverse proxy for the web and then selects a signal, a sequence of bits, for example, “1001”. To embed the signal into the target traffic, the attacker manipulates the web data at the reverse proxy, and carefully chooses the data size to embody bit 1 or 0. In this way, the signal will be embedded into the target traffic transmitted to the victim client via Tor. An accomplice of the attacker sniffs the traffic at the client side and determines whether that client has received the traffic embedded with the same signal.

As shown in Figure 4, the workflow of the packet size based attack is illustrated below.

Step 1: Modulate target traffic. In order to encode bit “0” of a signal, the attacker changes the reverse proxy’s buffer size as 498 bytes and forwards the data to the exit router. Recall that the original data size of one cell is 498 bytes as shown in Figure 2. Consequently, when the attacker writes 498 bytes into the connection to the exit router, the data will be exactly packed into one cell at the exit router.

In order to encode the “1” bit of original signal, the attacker varies the buffer size as 2444 bytes and transmits the data to the exit router. After receiving 2444 bytes, the exit router can pack the data into five cells. In general, the MTU is 1500 bytes, the IP header and TCP header are 20 and 32 bytes, respectively, and then the TCP payload should be 1448 bytes. The data size of 2444 bytes for “1” bit exceeds the MTU. Therefore, it is packed into two IP packets. One packet has TCP payload of 1448 bytes and the other has the TCP payload of 996 bytes. The first packet will be packed into three cells at the exit router, i.e., two cells with 498 bytes and one cell with 452 bytes. To ensure the equal-sized cells, Tor pads the data with 452 bytes to 498 bytes and forms one cell. The second packet will be split into two cells of exactly 498 bytes. Therefore, we obtain exact five cells. In this way, we map the signal into the desired number of cells. Then these cells are repacked into IP packets and transmitted to the client. However, due to the design of Tor queue process, MTU and network congestion, the data carrying the signal will be split or combined at the application layer and network layer. Reliable encoding mechanisms are required to deal with network dynamics. This issue is addressed in Section III-B.

It can be seen that if the data for consecutive bits are sent too close to each other, they could be merged at Tor routers. Therefore, we need to insert a marginal delay between consecutive bits. How to select the delay will be addressed in Section III-B.

Step 2: Demodulate the target traffic. An accomplice of the attacker at the client side sniffs (e.g., sniffing is easy in wireless networks) the encrypted traffic transmitted to the client at the entry router and records the packet size. The attacker can deduce the number of cells in a packet from the packet size. Eventually, the attacker can decode the signal in terms of the number of cells. We will discuss the format of the packets in Section III-B. By using our developed recovery mechanism, the attacker is able to extract the signal by decoding packets whose size embeds signal bits. In this way, the attacker confirms that the client accesses the targeted web via Tor.

B. Issues and Solutions

Due to the network dynamics, MTU and the processing capacity of Tor, the signal embedded in the traffic may be distorted. This affects the effectiveness of the attack. We address these issues below.

1) *Signal Distortion:* The major factors which lead to the signal distortion and affect the effectiveness of the attack are listed below.

Packet split: Refer to the cell processing at the onion router in Figure 3 introduced in [5]. Once the read event is called, cells are pulled from the TLS buffer into the input buffer at

a Tor router. Then the first cell can be pulled from the input buffer into the queue. If the output buffer is empty, this cell in the queue will be written into the output buffer promptly. Otherwise, it will wait in the queue. Then the other cells in the input buffer will be pulled into the queue one by one and wait for the coming write event. Once the write event is called, it will write the cell in the output buffer into the corresponding TLS connection and flush other cells from the queue to the output buffer. Then a subsequent write event will write cells in the output buffer into the TLS connection.

Because of such cell processing, a packet can be split at Tor routers. Consequently, five successive cells at the exit onion router will be split as one cell and four successive cells. Figure 5(a) and Figure 5(d) illustrate the structure of IP packets for one cell and four cells, respectively. In Figure 5(a), the IP packet with one cell consists of an IP header, a TCP header, an empty TLS application record and a TLS application record of enveloping one cell. Note that an IP packet cannot be packed with four cells because of the MTU (1500 bytes). The four cells will be split into two packets as shown in Figure 5(d). Since the default Tor path length is three, five successive cells will pass through three different onion router. Consequently, these five successive cells may be separated in the input buffer of three distinct routers. The five cells can be separated as three single cells and two successive cells. That is why we choose five cells for “1” bit. Otherwise, a smaller number of cells for “1” bit may cause all successive cells packed into individual small packets, which imply multiple “0”s and leads to decoding process hard or impossible.

The cases of packing cells into packets at Tor routers can be very complicated and listed below.

- One case is that the first packet with 1448 bytes of payload within 2444 bytes data for bit “1” can be split into three cells, while the second packet with 996 payload bytes can be split into two cells. If the network is congested between the exit router and middle router, these two cells may not arrive at the next router promptly. The first three cells will be transmitted and can be split twice when they pass through the middle router and entry router because of Tor’s cell processing procedure. Therefore, the 2444 bytes data may generate three single cells and two successive cells. Figure 5(b) illustrates the structure of the IP packet packed with the successive two cells.
- Another case is that the first three cells may be separated as one single cell and two consecutive cells. The subsequent two cells may be combined with the previous two cells. Then the four consecutive cells may be separated at the entry router as one single cell and three consecutive cells. Figure 5(c) illustrates the structure of IP packet, which is packed with three cells. Eventually, we obtain two single cells and three consecutive cells.
- The other cases are related to whether the second packet is split or not. Given the conditions where first packet may be separated as one single cell and two consecutive cells and the second packet may be split or not, we have extra more cases.

According to the analysis above, we can obtain 5 cases for bit “1”. We denote these cases as 1-1-1-2, 1-1-3, 1-4, 1-2-2,

and 1-2-1-1.

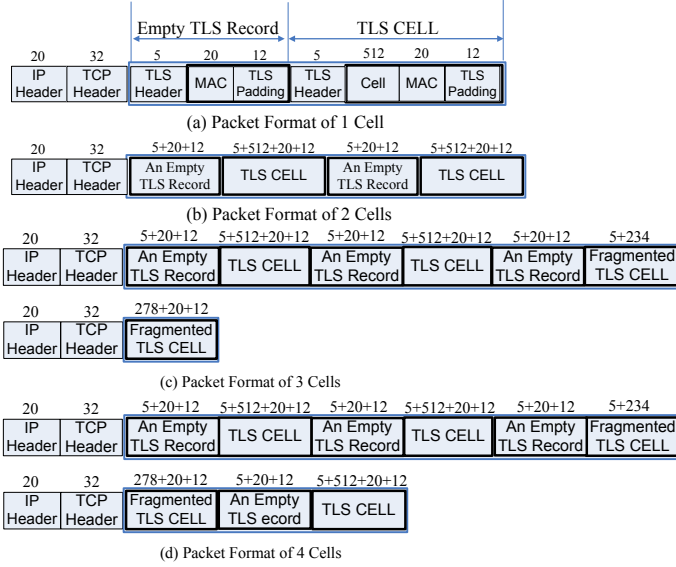


Fig. 5. Packet Format at the Network Layer

Cell merge: The cells carrying the signal can be merged with the fore-and-aft cells at the onion router. If the network is congested between two routers or between the router and client, the cells at the router cannot be transmitted promptly. Then the cells waiting in the queue at the router may be combined with the subsequent cells. The signal is then distorted. For example, to encode a signal “10”, the reverse proxy transmits a chunk of data of 2444 bytes and a subsequent chunk of data of 498 bytes. Because of MTU, the first data is split into 1448 bytes and 498 bytes, which are packed into two packets, respectively. The two packets then arrive at the exit router. If the load of the exit router is heavy, it can’t process the data promptly. The data can be split into cells, which wait in the queue. Then the second packet with 498 bytes data arrives at the exit router. This packet will be packed into one cell. This cell may be pulled into the queue and merged with the previous cells waiting in the queue. All of these cells in the queue are written into the TLS buffer once the write event is called. Hence, the cells carrying the signal “10” are merged and confuse the signal recognition based on packet size.

Packet retransmission: Due to the network dynamics between the entry router and the client, the packets may be retransmitted. This makes the signal recovery complicated as well.

As we can see, packet split, cell merge and packet retransmission will affect the effectiveness of the attack. To address these issues, we present several solutions below.

2) *Selection of Delay Interval:* In order to resist various interference, we consider to add additional delay between the data chunks carrying signal bits at the reverse proxy side. Denote T_i as the time of the i^{th} data arriving at the onion router. Denote T_p as the processing time of the i^{th} data at the onion router. A packet may be merged with the subsequent packet if

$$T_i + T_p > T_{i+1}. \quad (1)$$

Denote the delay interval added by the attacker as I , then

$$T_{i+1} = T_i + I. \quad (2)$$

Packets carrying two successive bits will not be merged if

$$T_i + T_p < T_{i+1} \quad (3)$$

$$T_i + T_p < T_i + I \quad (4)$$

$$I > T_p. \quad (5)$$

In order to prevent the signal distortion discussed above, the attacker needs to select a delay interval larger than the onion router’s processing time. Because the onion router’s processing time is dynamic, there are some tradeoffs of selecting delay interval: with the increase of delay interval, attack accuracy will increase and attack efficiency will reduce.

3) *Signal Recovery Mechanism:* To deal with packet merge, split and retransmission, we design a signal recovery mechanism to extract the signal from target traffic. The basic idea is illustrated below.

- First, we remove retransmitted packets between the entry router and client based on TCP sequence number.
- Second, since the attacker at the client side knows the original signal, she can recover the packets for bit “1” based on the known patterns, including 1-1-1-2, 1-1-3, 1-4, 1-2-2 and 1-2-1-1. Specifically, if an original bit is bit “1”, the attacker looks for such packet patterns in the subsequent packets. Hence, the attacker will not consider a single cell for bit “1” in the pattern as bit “0”.
- Third, to process the packets carrying a merged signal, we sum up the size of all merged packets. Denote the total packet size for the merged signal as Z . Then we try to find integers x and y such that $Z = 512x + 37y$, where 512 is the size of one cell and 37 is the aggregated size of the TLS header (5 bytes), MAC (20 bytes) and TLS padding (12 bytes). Note that the size of an empty TLS application record is also 37 bytes. In this way, we can calculate the number of the merged cells. According to the original signal and the number of calculated cells, the attacker can recover the signal ultimately.

C. Discussion

1) *Attack Accuracy:* Detection rate is defined as the probability that an original signal is recovered. It is affected by delay interval I . Denote the detection rate as $P(I)$. According to the analysis results in [5], the detection rate $P(I)$ is a monotonous increasing function in terms of the delay interval I . Therefore, the larger the delay interval we choose, the higher the detection rate becomes. This result is also validated by our real-world experimental data in Section IV. Denote the false positive rate as $P_{F,n}$ for recognizing a n -bit signal. As mentioned in [5], false positive rates decreases as the original signal length n gets longer.

2) *Countermeasures:* Inserting dummy packets into the original traffic can defend against the attack proposed in this paper to some extent. Nevertheless, it is commonly believed that dummy traffic will incur large overhead to both Tor routers and the client, dramatically degrading the performance of anonymous communication. To overcome the overhead, one possible alternative is selective padding similar to those in [6]. However, it is an open question regarding the gains that Tor can get and how much overhead it incurs to Tor. We leave such investigation as our future work.

IV. EVALUATION

In this section, we use real-world experiments to demonstrate the feasibility and effectiveness of the packet size based attack against Tor. All the experiments were conducted in a controlled manner over Tor and we experimented on TCP flows generated by ourselves to avoid legal issues.

A. Experimental Setup

Figure 6 illustrates the experiment setup. We deploy a malicious web site and a reverse proxy on Campus A. The web server and reverse proxy are installed on a single computer. The web server is Apache/2.2.11 (Linux) and the reverse proxy is Pen [14]. Two other computers are deployed on Campus B. All computers are running Fedora Core 11 operating system. One computer acting as a client is connected to a wireless access point and its traffic is not encrypted over the network. The other computer is used as a sniffer to record the size of packets from the entry router to the client computer.

We modified the code of the reverse proxy *Pen* to manipulate the packet size and implemented the encoding algorithm. For verification purposes, we use the client to download the file from the server. By configuring the reverse proxy *Pen*, we mapped the reverse proxy port 8080 to the HTTP server port 80. Hence, the reverse proxy can forward the packets for the web server. At the client side, the downloading software is the command line utility *wget*. By configuring *wget*'s parameters of *http_proxy* and *ftp_proxy*, we let *wget* download files through Privoxy, the proxy server used by Tor. By using the Tor configuration file and manipulatable parameters, such as *EntryNodes*, *ExitNodes*, *StrictEntryNodes*, and *StrictExitNodes* [13], we let the client choose both the desired entry and exit routers along the circuit. By modifying the Tor code, the middle router can be fixed as well. Then the Tor client will intend to setup circuits through the designated exit onion router (*OR3*), middle onion router (*OR2*) and entry onion router (*OR1*) shown in Figure 6.

B. Experimental Results

To validate the accuracy of the attack, we let the client download the file 30 times. At the reverse proxy, we generate a sequence of random signal of length 100 bits. When the target web traffic arrives at the reverse proxy, we vary the read buffer and embed a signal into the target traffic. At the client side, Sniffer records the packet size by removing the MAC (IEEE 802.11) header. Then the detection mechanism proposed in Section III-B is used to recognize the signal from the sequence of packet size.

To evaluate the false positive rate of the attack, we let the client download a file 30 times via Tor. However, no signal is embedded into the traffic at the reverse proxy. We refer to the traffic without signal as clean traffic. We then use the same detection mechanism proposed in Section III-B to detect 100 random signal from the clean traffic and calculate the false positive rate.

Figure 7 illustrates the relationship between the detection rate and the delay interval. Denote the probability that our signal is detected as p . If the victim traffic is marked by our signal m times, the probability that the victim is determined can be calculated as $1 - (1 - p)^m$. From Figure 7, we have

a few observations: (i) The detection rate increases dramatically when the delay interval increases. The detection rate approaches over 90% when the delay interval is 400ms and $m = 4$. (ii) With increasing m , the detection rate considerably increases. (iii) The false positive rate is less than 4% in all cases. These results validate that the attack using packet size can degrade the degree of anonymity that Tor promises.

Figure 8 illustrates the relationship between the detection rate and the delay interval, as well as the length of a signal when $m = 3$. Figure 8 shows that the detection rate will decrease while the signal length increases. From this figure, we know that only tens of packets is needed for our packet size based attack to achieve high detection rate. At the same time, the false positive is very low. This observation confirms that the attack is highly effective and can compromise the anonymous web downloading via Tor.

V. RELATED WORK

Traffic analysis is a common means to degrade communication privacy. There are a large number of related works on traffic analysis. We only review the most related ones because of the space limit.

Existing traffic analysis can largely be divided into two categories: passive traffic analysis and active watermarking techniques. Passive traffic analysis techniques have shown that the attacks record the traffic passively and identify the similarity between server's outbound traffic and client's inbound traffic [6]. For example, Zhu *et al.* [7] proposed the scheme of using mutual information for the similarity measurement. Levine *et al.* [6] investigated a cross correlation technique for the similarity measurement. Other recent research have shown that the attackers can infer sensitive information from the encrypted network traffic by examining patterns in terms of the sizes of packet and its timing [15], [17], [16]. Liberatore and Levine [15] examined the packet sizes of HTTP traffic transmitted over persistent connection or tunneled via SSH port forwarding can statistically identify the web pages. Wright *et al.* [17] investigated the statistical distribution of packet sizes in encrypted Voice over IP (VoIP) connections and identified the language spoken based on the distribution in each conversation. Later work of Wright *et al.* [16] also investigated how an eavesdropper could identify spoken phrases in encrypted VoIP.

Active watermarking techniques intend to embed specific secret signal into the target traffic. Such techniques can reduce the false positive rate significantly if the signal is long enough and does not require massive training study of traffic cross correlation as required in passive traffic analysis. For example, Wang *et al.* [18] proposed an active watermarking scheme that was robust to random timing perturbation. Overlier *et al.* [8] studied a scheme using one compromised mix router to identify the "hidden server" anonymized by Tor [1]. Murdoch *et al.* [9] investigated the timing-based watermark attack on Tor by using some compromised Tor routers. Kiyavash *et al.* [19] proposed a multi-flow approach detecting the interval-based timing watermarks [20], [2]. Note that timing based watermarks require much more traffic than packet size based attacks as we investigated because maintaining accurate timing is a daunting task and requires reasonable amount of traffic for statistical analysis and watermark accuracy. Yu *et al.* [3]

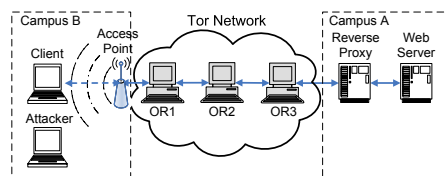


Fig. 6. Experiment Setup

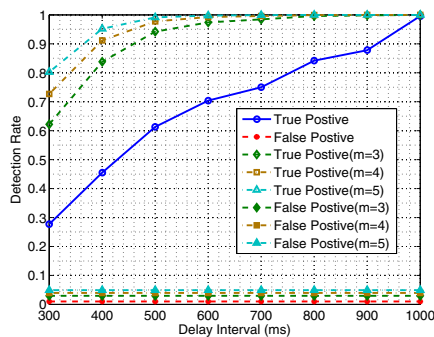


Fig. 7. Detection Rate v.s. Delay Interval

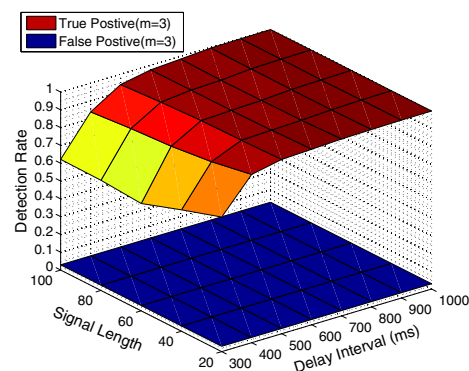


Fig. 8. Detection Rate v.s. Delay Interval and Signal Length

proposed a flow marking scheme based on the direct sequence spread spectrum (DSSS) technique. Ling *et al.* [5] proposed the cell counting based attack against Tor. However, that attack assumes that both *OR3* and *OR1* over the user's circuit are controlled by the attacker.

VI. CONCLUSION

In this paper, we investigated a packet size based attack to degrade the anonymity service provided by Tor. In particular, an attacker can manipulate the size of data into Tor in order to embed a signal into target traffic. An accomplice of the attacker at the client side recognizes the embedded signal using our developed recovery mechanism, and confirms the communication relationship among users. Via extensive real-world experiments on Tor, the effectiveness and feasibility of the attack is validated. Our data clearly demonstrates that equal sized cells at the application layer can't thwart an attacker who exploits the network layer packet size to degrade the anonymity service provided by Tor. Due to Tor's fundamental design, defending against this attack remains a very challenging task that we will investigate in our future research.

ACKNOWLEDGMENT

This work was supported in part by NSFC under grants 60903162, 60903161 and 90912002, and by USA NSF grants 0942113, 0958477, 0943479 and 0907964, National Key Basic Research Program of China under grant 2010CB328104, China National Key Technology R&D Program under grant 2010BAI88B03, China Specialized Research Fund for the Doctoral Program of Higher Education under grant 200802860031, Jiangsu Provincial Natural Science Foundation of China under grant No.BK2008030, and by Jiangsu Provincial Key Laboratory of Network and Information Security under grant BM2003201. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor agencies.

REFERENCES

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

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

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

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

[5] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A new cell counter based attack against tor," in *Proceedings of 16th ACM Conference on Computer and Communications Security (CCS)*, November 2009.

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

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

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

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

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

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

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

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

[14] "Pen," <http://siag.nu/pen>, 2010.

[15] M. Liberatore and B. N. Levine, "Inferring the source of encrypted http connections," in *Proceedings of the ACM conference on Computer and Communication Security (CCS)*, October 2006.

[16] 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 *Proceedings of the 2008 IEEE Symposium on Security and Privacy (S&P)*, May 2008.

[17] 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 *Proceedings of the 16th Annual USENIX Security Symposium (Security)*, August 2007.

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

[19] N. Kiyavash, A. Houmansadr, and N. Borisov, "Multi-flow attacks against network flow watermarking schemes," in *Proceedings of the 17th USENIX Security Symposium (Security)*, July/August 2008.

[20] Y. J. Pyun, Y. H. Park, X. Wang, D. S. Reeves, and P. Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, May 2007.