De-Anonymizing and Countermeasures in Anonymous Communication Networks

Ming Yang, Junzhou Luo, Zhen Ling, Xinwen Fu, and Wei Yu

ABSTRACT

To fulfill global requirements for network security and privacy, anonymous communication systems have been extensively investigated and deployed over the world to provide anonymous communication services for users. Nonetheless, diverse de-anonymizing techniques have been proposed to compromise anonymity and impose a severe threat to anonymous communication systems. In this article, we classify the existing de-anonymizing techniques and provide an overview of these techniques. In addition, corresponding countermeasures are studied to mitigate the risks posed by these de-anonymizing techniques.

INTRODUCTION

With rising concerns about privacy and security, Internet users employ numerous ways to encrypt their network traffic. According to the recent statistics of data networks in [1], encrypted traffic in North America has doubled since a year ago while more than quadrupling in Europe and Latin America. Currently, a number of people not only employ traditional encryption methods (e.g., SSL/TLS — Secure Sockets Layer/Transport Layer Security) to preserve the privacy of content in traffic, but also leverage anonymous communication networks to further protect their communication privacy and security.

To provide comprehensive anonymous communication service, researchers developed diverse anonymous communication systems (e.g., the onion routing based system Tor). In light of anonymous applications, anonymous communication systems can generally be categorized into two groups: message based (i.e., high-latency) and flow based (i.e., low-latency) systems.

Email and e-voting are the classic message-based anonymity applications, and have been well studied over the past decade. Because of the increasing need for anonymity over prevalent applications (e.g., web browsing and instant messaging), flow-based anonymity systems have been extensively studied and deployed around the world.

Various low-latency anonymous communication systems have been developed and deployed. By using a searching service, a list of free HTTP or SOCKS proxies can easily be found around the Internet. Additionally, one of the most popular anonymous communication systems (Tor) provides anonymous communication services for hundreds of thousands of Internet users and carries terabytes of traffic each day. As of September 2014, there were more than 6000 Tor routers voluntarily deployed around the world to contribute their bandwidth to the entire Tor network.

A number of de-anonymizing techniques have been investigated to compromise users' communication anonymity. Particularly, traffic-analysis-based de-anonymizing techniques are the primary threats to anonymous communication systems. In this article, we provide an overview of existing de-anonymizing techniques and countermeasures in flow-based anonymous communication systems. To be specific, we first model two categories of anonymous communication systems and introduce their basic anonymizing techniques. We then categorize the de-anonymizing techniques into four groups from different perspectives and elaborate on these techniques from each groups. Existing and possible countermeasures for these de-anonymizing techniques are also studied to improve the anonymity service provided by anonymous communication systems.

The rest of this article is organized as follows. We first introduce two types of classic anonymous communication models. Then we categorize the de-anonymizing techniques and introduce them in detail. Next the corresponding countermeasures are investigated. Finally, we conclude this article.

ANONYMOUS COMMUNICATION MODEL

In this section, we briefly introduce two categories of flow-based anonymous communication models in terms of the length of

Ming Yang, Junzhou Luo, and Zhen Ling are with Southeast University.

Xinwen Fu is with the University of Massachusetts Lowell.

Wei Yu is with Towson University.

0163-6804/15/$25.00 © 2015 IEEE

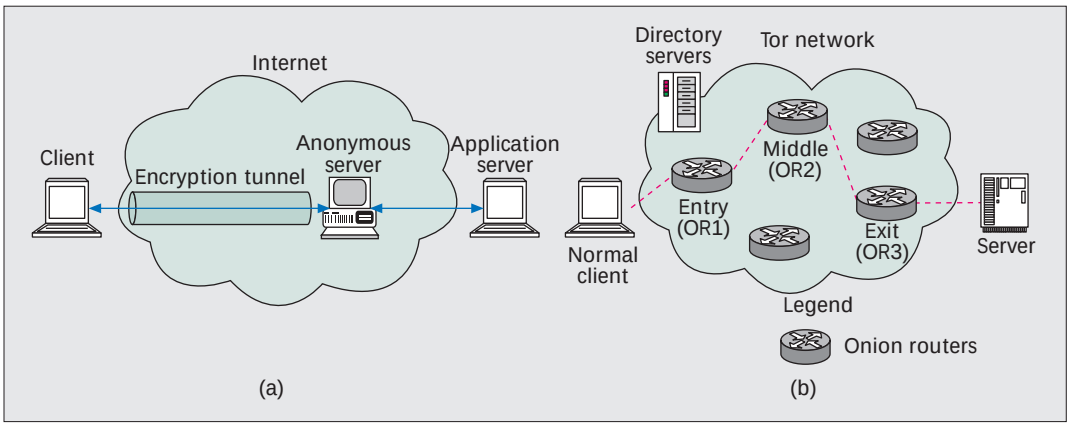

Figure 1. Anonymous communication model: a) single-hop anonymous communication model; b) multihop anonymous communication model (Tor)to compromise users' communication anonymity.

Currently, VPNs and encrypted tunnels to single-hop proxies are typical single-hop anonymous communication systems. These types of systems provide better performance. Nonetheless, a compromised single-hop proxy will result in the exposure of all clients' traffic through the proxy.

anonymous communication links, including single-hop and multihop anonymous communication models.

Single-Hop Anonymous Communication Model

The single-hop anonymous communication model has been widely applied over the Internet due to its effectiveness and efficiency. Broadly speaking, there are three basic components in this model, including a client, an anonymous server, and an application server. Figure 1a illustrates a traditional single-hop anonymous communication model. A client first installs anonymous service software. Then, by using the installed software, the client can establish an encryption tunnel to a specific anonymous server and delivers application data associated with the user to the anonymous server over this tunnel. The anonymous server decrypts the data and forward it to the destination application server. The application server supports Transmission Control Protocol (TCP) applications (e.g., FTP servers and web servers) and receives the data from the anonymous server. After that, the corresponding data is transmitted to the server. Because the application server cannot identify the real IP address of the client, the client can anonymously communicate with the remote server.

Currently, virtual private networks (VPNs) (e.g., OpenVPN and CiscoVPN) and encrypted tunnels to single-hop proxies (e.g., OpenSSH) are typical single-hop anonymous communication systems. These types of systems provide better performance. Nonetheless, a compromised single-hop proxy will result in the exposure of all clients' traffic through the proxy. To overcome this issue, the multihop anonymous communication model has been developed to mitigate this risk.

Multihop Anonymous Communication Model

Again, to address the security issue of the single-hop anonymous communication model, multihop anonymous communication systems such as Tor and JonDonym have been proposed. Because Tor is the most popular multihop anonymous communication system, we take it as an example to depict the organization of a multihop anonymous communication system. Broadly speaking, a Tor network consists of four components: a client, onion routers, directory servers, and server. Figure 1b illustrates the basic architecture of a Tor network. A client needs to install Tor software (i.e., *Oni n P* , OP) to pack the data from the client into a Tor *cell*, which is a basic transmission unit in the Tor network. *Oni n e* (ORs) are used to relay data between clients and servers. The directory servers collect all of the information associated with the ORs, including IP address, port, public key, and so on. Servers provide TCP application services such as web and FTP services.

To anonymously communicate with the remote server through Tor, the client builds a multihop path using a source routing mechanism and communicates with the remote server through the established path in the Tor network. First, the client downloads all of the OR information from the directory servers and selects several high-performance ORs. The number of selected ORs in a path is denoted as the path length. The default path length is three, which is hard-coded in the Tor client software. The path is also referred to as a *ci - c i* in the Tor network. Upon choosing the appropriate ORs, an OP will establish a one-hop circuit to the first OR (i.e., entry OR) and negotiate a symmetric key. After that, it will extend this tunnel to the following two ORs, referred to as the middle and exit ORs, respectively, and negotiate symmetric keys with them. Once the circuit is completely built, multi-TCP streams from the client are multiplexed into this circuit.

It is worth noting that the multihop anonymous communication model is more secure than the single-hop anonymous communication model. In the case of Tor, because any respective OR in the circuit cannot link the client and server, the system can provide better anonymous service to the users. Nonetheless, the data from users is relayed through more nodes in a multihop anonymous system, which definitely increases the end-to-end latency.

End-to-end attacks explored in existing work mainly focus on correlation attacks to confirm the communication relationship between senders and receivers. In contrast, single-end attacks focus on fingerprinting-based attacks to identify the victim's accessed web page.

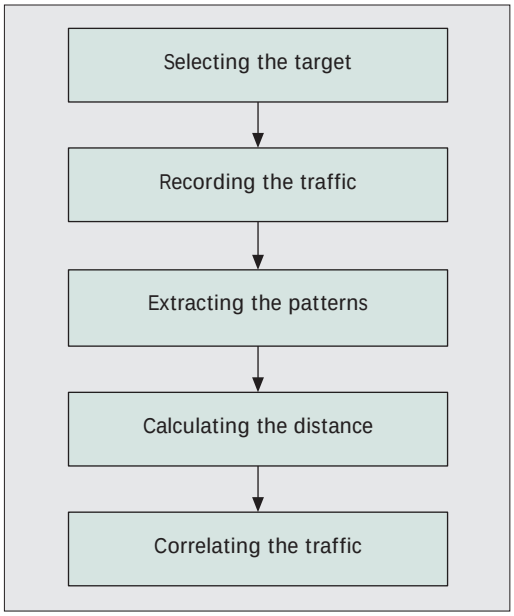

Figure 2. Workflow of end-to-end passive attacks.

De-Anonymizing Techniques to Compromise Users' Communication Anonymity

In this section, we first categorize de-anonymizing techniques into two groups from two different perspectives. We then elaborate on these techniques individually.

Category of De-Anonymizing Techniques

According to existing network-traffic-analysis-based de-anonymizing techniques, we introduce two dimensions of attacks.

- Passive and active attacks: The adversary can passively monitor the victim's traffic or actively manipulate the traffic.
- Single-end and end-to-end attacks: The adversary conducts attacks by monitoring or controlling related devices at either the sender or receiver side, or at both the sender and receiver sides.

End-to-end attacks explored in existing work mainly focus on correlation attacks to confirm the communication relationship between senders and receivers. In contrast, single-end attacks focus on fingerprinting-based attacks (e.g., website fingerprinting attacks) to identify the victim's accessed web page. In the following, we discuss end-to-end attacks and single-end attacks in detail.

End-to-End Attacks

End-to-end attacks are designed to correlate the communication relationship between clients and servers using either passive or active attack methods. To carry out attacks, the adversary should control or monitor the devices (e.g., routers or Tor entry and exit nodes) at both the sender and receiver sides. In the following, we discuss passive and active end-to-end attacks, respectively, and then summarize the advantages and disadvantages of these attacks.

End-to-End Passive Attack — The object of the end-to-end passive attack is to record traffic passively and evaluate the similarity between the sender's outbound traffic and the receiver's inbound traffic based on statistical measures. Figure 2 illustrates the basic workflow of end-to-end passive attacks. This type of technique can exploit traffic features (e.g., packet counter, traffic pattern correlation, timing correlation). For example, the adversary can simply count the number of outgoing packets in several time intervals at the output link of the sender and then count the number of arrival packets in the same time interval at the input link of the receiver. Then a distance function can be applied to compute the distance between these two links in terms of traffic features.

The primary advantage of end-to-end passive attacks is stealth because the traffic will only be monitored. Nonetheless, the true positive rate is low, while the false positive rate is high. Accordingly, the adversary needs a sufficient amount of time to observe the traffic and discover traffic pattern similarities between senders and receivers in order to reduce the number of errors associated with the attack. In addition, to improve the true positive rate and reduce the false positive rate, end-to-end active attacks have been proposed to manipulate traffic in order to generate a desired signal.

End-to-End Active Attack — The basic idea of this attack is that the adversary can manipulate traffic at the sender or receiver side by embedding a special signal in the victim's traffic. Then the traffic at the receiver or sender side is monitored in order to recognize the signal and confirm the communication relationship between the sender and the receiver. This class of attacks is also referred to as *watermarking-based attack* [2].

Because the adversary can exploit various features of distinct layers to embed watermarking into network traffic, we present these attacks from three different layers: *network layer*, *data-link layer*, and *application layer*.

At the network layer, the adversary can exploit such features as the traffic rate [2], packet delay interval [4], and packet size to embed a signal into target traffic. For example, Yu *et al.* [2] proposed that an adversary could interfere with traffic from a sender and shape its traffic rate pattern. In this way, an invisible direct sequence spread spectrum (DSSS) signal can be embedded in the traffic. Then the embedded signal along with the traffic transmitted through the anonymous communication network arrives at the receiver. After that, the adversary can recognize the signal and compromise the anonymity between the sender and the receiver.

Wang *et al.* [4] investigated packet delay interval centroid-based watermarking techniques. Assume that the arrival distribution of packets in a time interval $[0, T)$ is uniform. By having the adversary intentionally delay each packet within this interval T, packets can uniformly exhibit values in the range $[a, T)$ so that the mean of packet arriving times in this time interval is $T + a/2$. In order to embed the signal in the traffic, the adversary first chooses two

groups of time intervals. Denote the two original groups as A and B and the delayed groups as A' and B'. To encode a 1 bit, the packets in the time interval of group A are carefully delayed, and the mean of these two groups can be derived as

$$E(A') - E(B) = \frac{T+a}{2} - \frac{T}{2} = \frac{a}{2}. \quad (1)$$

To encode a 0 bit, the packets in the time interval of group B are altered, and the adversary can adjust the mean of these two groups as

$$E(A) - E(B') = \frac{T}{2} - \frac{T+a}{2} = -\frac{a}{2}. \quad (2)$$

By adjusting the time interval centroid, a series of binary signal bits can be embedded into the traffic.

In addition, an adversary can vary the packet size to embed a signal into the victim's traffic. For example, the adversary controls a web server and manipulates the size of the response HTTP packets. A specific packet length can be mapped into a single hex bit. By altering the length of several packets, the adversary can encode a message into the traffic. Although the packet length is partially padded at the single-hop proxy, the adversary can still infer the packet length in order to recover the original signal at the client side and confirm the communication relationship between the client and the server. Additionally, to keep this attack invisible, the adversary needs to keep both the distribution and the self-similarity of the original packet size. To this end, the adversary needs to deliberately select appropriate packets and alter their sizes.

At the protocol layer, the watermarking attack can employ different protocol features of anonymous communication systems. For example, Ling *e al.* [5] deeply explored the communication protocol of Tor and discovered that Tor employs the counter mode of the Advanced Encryption Standard (AES-CTR) to encrypt and decrypt Tor cells. Consequently, each Tor node, including the Tor client in a circuit, maintains a local counter to synchronize the counter values with each other in order to correctly encrypt or decrypt the cells. Nonetheless, this attack exploits the feature of the counter synchronization mechanism in a multihop path, and disturbing the counter value at some node along this path incurs encryption/decryption failure of the Tor cell. To achieve this goal, the adversary first needs to control both the Tor exit and entry nodes, and then have the ability to operate the Tor cell at the entry node. Specifically, the adversary can replay, delete, or insert a cell to the target circuit at the entry node. Replaying a cell or inserting a faked cell will result in increasing the counter at both the middle and Tor exit nodes, whereas deleting a cell will decrease the counter. These operations can make the counter value at the middle and exit Tor nodes unsynchronized and cause cell decryption failure at the exit node. Because this type of decryption failure is fairly rare in a normal circuit, the adversary can use this unique feature to detect whether a manipulated cell passes through its controlled Tor exit node. Once this decryption failure is recognized at the exit node, the adversary at the entry node knows the source of the circuit, while the conspirator at the exit node learns the destination of the circuit. In this way, the adversary can trivially confirm the communication relationship between the source and destination of the circuit.

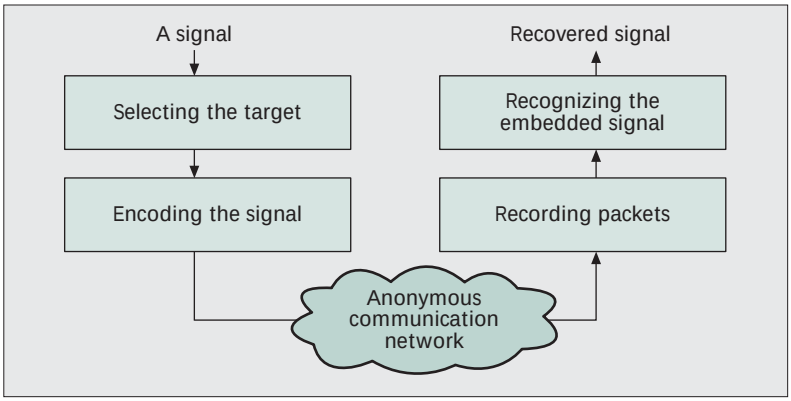

Figure 3. Workflow of end-to-end active attacks [3].

Additionally, the adversary can exploit the defects in data integrity verification in multihop anonymous communication systems [5]. For example, the Tor cell is encrypted in an onion-like fashion during the transmission in the circuit. Consequently, unlike the Tor client and exit node, which can obtain the plaintext of the cell to check the integrity of the data, entry and middle nodes cannot verify the integrity of the cell. If the adversary tampers with the content of the ciphertext of the cell at an entry node, it results in cell decryption failure at the exit node due to a lack of data integrity verification at each node along the entire multihop path. Likewise, the adversary at the exit node can leverage decryption failure as a signal to correlate the communication relationship between the sender and the receiver.

The adversary can also use the Tor protocol characteristic (i.e., the size of each Tor cell is equal [3]). The adversary can control the number of transmitted cells during each time slot at a Tor node in order to encode signal bits. Particularly, to encode a 1 bit, the exit node will collect three Tor cells in the circuit queue and flush them out into the circuit. In addition, a single cell will be sent into the circuit to encode the 0 bit. To avoid the problem of adjacent signals being merged together, a delay interval can be introduced between signals. When the cells that carry the signals arrive at the entry node, a sophisticated signal recovery algorithm can be applied to decode and recover the signals in terms of the number of the received cells in the circuit queue. Once the signal is detected, the communication relationship between the sender and the receiver can be correlated.

At the application layer, the adversary at the server side can inject special content into the victim's web response traffic in order to force the client to generate special traffic patterns as a signal. Then the adversary at the client side can observe this signal and confirm the communication relationship between the sender and the receiver. Specifically, once an

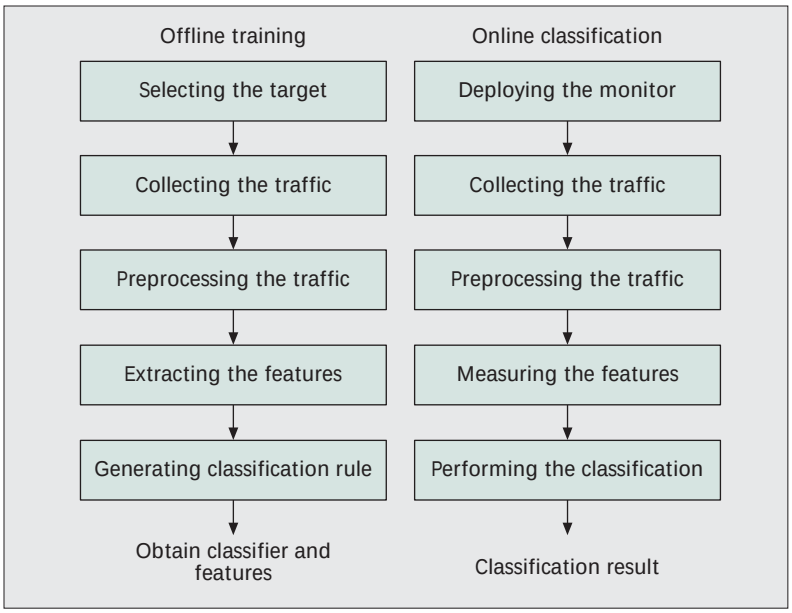

Figure 4. Workflow of single-end passive attacks.

adversary discovers the target web response traffic that passes through its exit node, malicious web links of empty images can be injected so that the browser at the client side will be forced to generate a specific traffic pattern to download these links [6]. Then the conspirator at the entry node will inspect the traffic to detect the desired pattern. Upon discovering the expected traffic pattern, the adversary can compromise the anonymity between the sender and the receiver. In this case, the adversary takes advantage of the HTTP application features to generate a signal. Moreover, similar techniques can be used to conduct such kinds of application layer attacks. For example, a piece of Javascript code can also be injected into the victim's web response traffic as a signal generator, which is executed at the victim's browser to generate signal traffic.

Single-End Attacks

The adversary who performs single-end attacks needs to control or monitor traffic passing through devices at the sending or receiving side so as to compromise the users' communication privacy and security. Single-end passive attacks extract the pattern of traffic, referred to as a fingerprint, and infer the content of traffic at the application layer (e.g., users' accessed websites). In addition, single-end active attacks can actively inject content into traffic at the application layer in order to force the client to directly send a signal to the adversary and expose the real IP address of the client. In the following, we elaborate on these attacks.

Single-End Passive Attack — The idea of single-end passive attacks is to monitor traffic between the victim and the anonymous proxy and identify the real accessed web pages by comparing a prospective traffic pattern with pre-collected web page fingerprints. This type of attack is also referred to as a Website Fingerprinting (WF) attack.

There are two phases of this attack:
- Offline training
- Online classification

Figure 4 illustrates the basic workflow of a single-end passive attack. In the offline training phase, the adversary first needs to select several websites of interest and set up the victim's environment to emulate the procedure of the victim's browsing activities. Then, the adversary will browse the websites one by one and collect the website traffic. Furthermore, the collected data should be preprocessed in order to remove noise. For example, the accuracy of the website fingerprinting attack will be affected because advertisement links on web pages are dynamic. Hence, by using some preprocessing strategies, this type of noise will be filtered. Additionally, the adversary should extract appropriate features from the preprocessed traffic. These features should be carefully selected to exhibit the most effective patterns, which are usually hidden in the traffic. According to existing attacks, various features have been explored to effectively conduct attacks, including packet length distribution, traffic volume, total time, traffic direction, packet length order, up/downstream bytes, bytes in traffic bursts [7], etc. Finally, the adversary chooses a proper classifier to generate a classification rule by using the collected sample data. In existing attacks, various classifiers have been investigated, including Bayes classifiers, multinomial naive-Bayes classifiers, Support Vector Machines (SVMs), decision trees, etc.

In the second phase, the adversary can record real traffic and launch an attack to identify the victim's accessed web pages. First, the adversary needs to deploy a monitoring tool and then silently collect the victim's traffic between the client and the anonymous proxy. After obtaining real traffic, the adversary will preprocess the traffic in order to remove noise. Moreover, the adversary measures the features from the processed traffic and performs the attack by using the classification rule to identify accessed web pages.

Single-End Active Attack — This type of attack actively inserts malicious code into non-encrypted traffic at the server side so that the code arrives and executes at the victim's host. This is done in order to bypass the installed client of an anonymous communication system and directly establish a connection to a malicious server. To this end, the adversary should control the non-encrypted link between the proxy and the remote server. For example, in the case of the Tor network, an adversary who controls a Tor exit node can arbitrarily inject or modify content of non-encrypted traffic. After assuming control of a non-encrypted link, the adversary can inject diverse software instances into the link, including Flash, Javascript, ActiveX Controls, and Java. Once these software types are executed in the browser, they will bypass the local proxy settings in the browser and directly create a connection to a specific remote server in order to expose the real IP address of the client. In addition, the adversary can take advantage of browser exploits to conduct this type of attack to compromise a victim's anonymity.

To prevent themselves from experiencing active single-end attacks, users should disable active content systems such as Flash, ActiveX Controls, Java, and Javascript to avoid malicious code executing in the browser. Alternatively, a transparent proxy can be deployed at the client side in order to ensure that all of the traffic is directed into the anonymous communication system.

C UNTERMEASURE

To mitigate the threat posed by de-anonymizing techniques, there have been a number of research efforts on developing various countermeasures to defend against these attacks. Broadly speaking, the countermeasures can be deployed from three perspectives: network layer, protocol layer, and application layer. In the following, we discuss these diverse countermeasures.

Because network traffic characteristics can be exploited to de-anonymize the communication between users, a basic idea of defense that can be used to thwart attacks at the network layer is to remove the features of traffic associated with users, including packet size distribution, packet order, traffic volume, traffic time, and so on. To be specific, *acke adding echni e* can be used to pad packet sizes in order to remove the packet length feature from features such as packet size and packet order. Intuitively, the size of each packet can be padded into the same size (e.g., maximum transmission unit, MTU). Additionally, various sophisticated strategies have been studied to effectively and efficiently pad the packet size [8]. To obfuscate the traffic time, delay can be intentionally added between each packet to increase the traffic time. Furthermore, *d mm affic echni e* can be applied to inject dummy packets into users' original traffic in order to obfuscate the traffic volume. In addition, *affic m hing echni e* can be used to vary current traffic patterns to look like other traffic patterns. For example, to thwart a website fingerprinting attack, the web server can first select a target page and then mimic the packet size distribution of that target web page. Generally speaking, defense techniques at the network layer are more general and can be used in various anonymous communication systems, although they can incur high transmission overhead.

At the protocol layer, protocol-level padding and dummy techniques can be used to hide traffic features associated with users. As a matter of fact, secure shell (SSH), TLS, and IPsec apply such protocol-level padding techniques to align plaintext with block cipher boundaries, thereby obfuscating the packet size to some degree. To further improve security, a random amount of padding can be selected [7]. Additionally, protocol-level dummy techniques can be used. For example, Tor does not commonly employ the functionality of padding cells for circuit-level padding purposes because it can significantly decrease the performance of the circuit. Protocol-level padding and dummy techniques could be designed to reduce the overhead incurred to some degree. Nonetheless, it should be carefully designed; otherwise, it could be used to conduct an MTU end-to-end active attack.

At the application layer, HTTP features and background traffic (i.e., decoy web pages) can be exploited to remove traffic features from user flows. For example, HTTP pipelining and HTTP ranges can be used to adjust both incoming and outgoing packet sizes [9]. Moreover, changing the order of the HTTP requests at the client side can vary the traffic pattern to some extent. To apply background traffic techniques at the application layer, a decoy web page can be silently loaded in the background while a user is browsing a target web page. This type of defense technique can only be used for some specific applications (e.g., HTTP) and cannot be widely applied for diverse applications [3].

Hybrid techniques can be deployed at the different layers to provide a comprehensive solu-

ap[8] (tiompt, send tiy current)1j 0.0493 3c 0.0007 TTTPact, spe of dsexte w T (fu

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under grants 61272054, 61402104, and 61320106007, China National High Technology Research and Development Program under Grant No. 2013AA013503, by U.S. NSF grants 1116644, 1350145, and CNS 1117175, Jiangsu Provincial Key Laboratory of Network and Information Security under grant BM2003201, and Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under grant 93K-9. Any opinions, findings, conclusions, and recommendations in this article be are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Sandvine, "Global internet phenomena report (1h 2014)," https://www.sandvine.com/downloads/general/global-internetphenomena/2014/1h-2014-global-internet-phenomenareport.pdf, 2014.
[2] W. Yu *et al.*, "DSSS-Based Flow Marking Technique for Invisible Traceback," *Proc. 2007 IEEE Symp. Security and Privacy*, Berkeley, CA, May 2007, pp. 18–32.
[3] Z. Ling *et al.*, "A New Cell-Counting-Based Attack Against Tor," *IEEE/ACM Trans. Net.*, vol. 20, no. 4, 2012, pp. 1245–61.
[4] X. Wang, S. Chen, and S. Jajodia, "Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems," *Proc. IEEE Symp. Security & Privacy*, Berkeley, CA, May 2007, pp. 116–30.
[5] Z. Ling *et al.*, "Protocol-Level Attacks Against Tor," *Computer Networks*, vol. 57, no. 4, Mar. 2013, pp. 869–86.
[6] X. Wang *et al.*, "A Potential HTTP-Based Application-Level Attack Against Tor," *Future Generation Computer System*, vol. 27, no. 1, Jan. 2011, pp. 67–77.
[7] K. P. Dyer *et al.*, "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail," *Proc. 2012 IEEE Symp. Security and Privacy*, San Francisco, CA, May 2012, pp. 332–46.
[8] X. Cai *et al.*, "A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses," *Proc. 21th ACM Conf. Computer and Communications Security*, Scottsdale, AZ, Nov. 2014, pp. 227–38.