Novel and Practical SDN-based Traceback Technique for Malicious Traffic over Anonymous Networks

Zhen Ling*, Junzhou Luo*, Danni Xu*, Ming Yang*, and Xinwen Fu‡
*Southeast University, Email: {zhenling, jluo, dannyxu, yangming2002}@seu.edu.cn
‡University of Central Florida, Email: xinwenfu@ucf.edu

***Abstract*—Diverse anonymous communication systems are widely deployed as they can provide the online privacy protection and Internet anti-censorship service. However, these systems are severely abused and a large amount of anonymous traffic is malicious. To mitigate this issue, we propose a novel and practical traceback technique to confirm the communication relationship between the suspicious server and the user. We leverage the software-defined network (SDN) switch at a destination server side to intercept target traffic towards the server and alter the advertised TCP window sizes so as to stealthily vary the traffic rate at the server. By carefully varying the traffic rate, we can successfully modulate a secret signal into the traffic. The traffic carrying the signal passes through the anonymous communication system and reaches the SDN switch at the user side. Then we can detect the modulated signal from the traffic so as to confirm the communication relationship between the server and the user. To validate the feasibility and effectiveness of our technique, extensive real-world experiments are performed using three popular anonymous communication systems, i.e., SSH tunnel, OpenVPN tunnel, and Tor. The results demonstrate that the detection rates approach 100% for SSH and Open VPN and 95% for Tor while the false positive rates are significantly low, approaching 0% for these three systems.**

***Keywords*—Anonymous communication systems, traceback technique, software-defined network**

I. Introduction

Various anonymous communication systems (e.g., Tor and VPN) have been developed and deployed around the world to protect users' communication privacy and fight against the Internet censorship [24]. For example, one of the most popular anonymous communication systems, Tor, employs more than 6000 Tor relay servers. It serves more than 2 million users and relays around 1 petabyte of traffic daily.

Unfortunately, the anonymous communication systems are wildly abused. For example, CloudFlare claims that 94% of Tor traffic toward the CloudFlare network is malicious [1]. Illegitimate users exploit Tor to send spams, download copyrighted or illegal materials such as child pornography [8], and even deploy botnet Command and Control servers (C&C) [12]. Black markets and C&C servers of ransomware (e.g., WannaCry) are deployed via Tor *hidden service* so as to avoid being discovered and taken down. Thus, it is of vital importance to have the capability of on-demand traceback of the malicious traffic transmitted from the server to the client in the Tor network. However, since the anonymous communication systems leverage one-hop or multi-hop proxy servers to establish an anonymous and encrypted tunnel so as to relay the users' traffic, the destination server can only observe the IP address of the last hop proxy server, making the tracebacking work complicated. Therefore, it is nontrivial to trace the communication between the user and the server.

To address this issue, active watermarking techniques [5], [6], [10], [13] are adopted to confirm the communication relationship between the user/client and server. The crucial idea of the techniques is to actively modulate a specific signal into the user's traffic at one communicating party and recognize the demodulated signal at the other side so as to confirm the communication relationship between the two communicating parties. However, the existing techniques highly rely on the special protocol features of different anonymous communication systems to design specific watermarking schemes that are hard to be widely deployed. Fortunately, the emerging software-defined network (SDN) provides us an opportunity to design a more general and robust traceback technique that can be applied for diverse anonymous communication systems as the control plane is decoupled from the data plane in networking equipment (e.g., switches and routers). OConnor *et al*. [16] leverage the SDN to trace the source of the APT traffic using stepping stones (i.e., the proxy servers) by labeling the Type of Service (ToS) field of the IP header in the kernel of each host and then detect the labeled packets at the SDN switch. However, a customized kernel should be installed in each host so as to achieve the accurate labeling-based traceback. Consequently, it can just be applied in a small controlled network environment.

In this paper, we propose a novel and practical SDN-based traceback technique to trace the malicious traffic over different anonymous communication systems including SSH tunnel, OpenVPN tunnel, and Tor. We design a new and more universal traceback technique and take full advantages of SDN to quickly and accurately confirm the communication relationship between the server and the client. To this end, we leverage the SDN controller that approaches a suspicious destination (e.g., server Bob) to intercept traffic towards the server. Then the advertised TCP window size of the packets is changed at the controller so as to force the server to change the traffic rate. In this way, we can modulate a randomly generated signal (i.e., a sequence of binary bits) by varying the server's outgoing traffic rate. To enhance the robustness of our signal, repetition error correcting codes are applied to modulate the original bits multiple times. Then the traffic carrying the signal traverses the anonymous communication system and arrives at the client (e.g., Alice). The SDN switch that approaches the client can record the timestamps and packet sizes of the traffic and demodulate the repetition bits. In light of the repetition scheme, we can derive the recovered signal and then determine if it is the original signal embedded at the server side SDN controller. We have implemented the SDN-based traceback technique and conduct extensive real-world experiments to validate its feasibility and effectiveness.

Our major contributions are summarized as follows.

- We leverage SDN to intercept the target traffic towards a suspicious destination and deliberately vary the advertised TCP window size of the traffic. In this

way, we can stealthily vary the outgoing traffic rate of the destination on demand so as to modulate a secret signal into the target traffic. We are the first to regulate the advertised TCP window size and embed a signal into traffic for traceback. This approach can be easily deployed over SDN.

- We carefully analyze TCP sliding window mechanism and calculate the minimum and maximum values of the regulatable advertised window size so as to exert minimal effects on the target traffic rate. Moreover, we employ repetition error correcting codes to enhance the robustness of our embedded signal.
- Our traceback technique is evaluated using three major anonymous communication systems including SSH tunnel, OpenVPN tunnel, and Tor. The real-world experimental results demonstrate that the detection rates can approach 100% for SSH tunnel and OpenVPN tunnel and 95% for Tor while the false positive rates for all the three communication systems are approximately 0%.

The idea of confirming the communication relationship between Alice and Bob can be generalized to perform full fledged traceback in a complicated network like Tor, where there are three hops between Alice and Bob in general. If SDN is pervasively deployed, the confirmation can be performed hop-by-hop starting at one side of the communication. Each time, the SDN is used to determine the next hop along the path until the other side of the communication is reached.

The rest of this paper is organized as follows. We introduce anonymous communication network and the software-defined network in Section II. Then we present the SDN-based traceback technique, including the basic idea and the detailed design of our system in Section III. In Section IV, we perform theoretical analysis on the selection of TCP window size and performance metrics. In Section V, we conduct extensive real-world experiments to demonstrate the feasibility and effectiveness of the traceback technique. We review related work in Section VI. Finally, we conclude this paper in Section VII.

II. Background

In this section, we briefly introduce the anonymous communication network and the software defined network.

A. Anonymous communication network

Diverse low-latency anonymous communication network systems are pervasively deployed around the world for protecting users' communication privacy and providing anti-censorship services. In light of the length of the anonymous communication connections, they can be categorized into two classes: single-hop and multi-hop anonymous communication network systems. In the single-hop anonymous network systems (e.g., SSH and OpenVPN), only one proxy server is used to relay a user's traffic to a destination server. Since the server can only observe the IP address of the proxy server, it cannot know the user's real IP address. However, the user's communication privacy can be exposed if the proxy server is compromised. In the multi-hop anonymous communication network systems (e.g., Tor), a user first communicates with the directory servers in the Tor network and downloads the information of Tor relay servers. Then the user chooses three relay servers and establishes a three-hop path hop by hop. Finally, she commands the last relay server along the Tor path to build a TCP connection to the destination server. In this way, the user can anonymously communicate with the server. Since the Tor relay servers in the path can only know the IP addresses of their adjacent relay servers in the path, one single compromised server can hardly confirm the communication relationship between the user and the server.

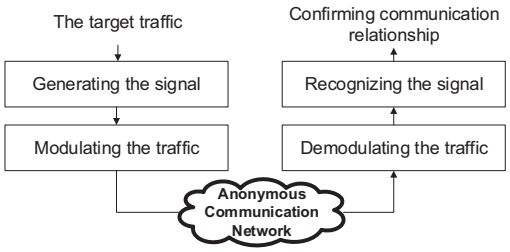

Fig. 1. Workflow of the SDN-based traceback technique

B. Software Defined Network

The software-defined network provides network programmability for dynamically managing and controlling the network via open APIs and protocols. The architecture of a software-defined network consists of three layers, including the data plane, control plane, and application plane. The data plane is composed of networking devices, e.g., SDN switches, used for forwarding network traffic. The flow entries in flow tables are stored at the switches and work as the forwarding rules. These flow entries are configured by the SDN controller. The control plane is composed of SDN controllers that control a set of networking devices in the data plane. The SDN controllers execute the requests from the SDN applications and make the low-level network services, e.g., network topology, available to the application plane via open APIs. The application plane is composed of a set of applications that can access the low-level network services provided by the SDN controllers. The applications can send high-level policies to the control plane that implements the policies as flow entries and sends them to the flow table on the networking devices. Since the SDN controllers are responsible for configuring all of the flow entries, they play a crucial role in managing and controlling the network.

III. SDN-based Traceback Technique

In this section, we first present the basic idea of the SDN-based traceback technique. Then we elaborate on the crucial steps of our technique.

A. Basic Idea

Our goal is to determine the communication relationship between the Alice (client) and the Bob (server) who are communicating with each other through anonymous network systems (e.g., VPN or Tor). It is assumed that the SDN switches are pervasively deployed over the Internet in the near future. Ideally, if the suspicious traffic traverses all of the SDN switches controlled by ourselves, we can trace back the traffic hop by hop to discover all servers in the entire path. However, it is resource-consuming to completely discover all the proxy servers. Instead, just by controlling two SDN switches that respectively approach the client and the server, we can observe the traffic from the client and the server via these two SDN switches. The SDN switch that observes the traffic from the server Bob is referred to as the *server-side*

SDN switch and the one that observes the traffic from the client Alice is referred to as the *client-side* SDN switch. Then we can design traceback techniques in an attempt to confirm the communication relationship between the Alice and the Bob. The traceback can be performed on either the server-side SDN switch or the client-side SDN switch. In the rest of the paper, we focus on the traceback just using the two SDN switches while our techniques can be used to perform the traceback hop-by-hop. In addition, it is assumed that the traceback is initiated at the server-side SDN switch.

Figure 1 illustrates the workflow of our SDN-based traceback technique. We first select target traffic at the server-side SDN switch and leverage the server-side controller to send a flow entry to the switch so as to force the switch to forward the traffic to the controller. Then we modify the TCP packets towards the server by changing the advertised TCP window size in the TCP packet header in order to alter the send window size of the server Bob. According to the TCP protocol, Bob has to change the send window size in light of the advertised TCP window size from the proxy servers (e.g., Tor relay servers and SSH proxy servers) so as to vary the server's traffic rate. As a result, we can generate a signal (i.e.,a series of signal bits) and modulate it into the traffic by carefully regulating the traffic rate sent from Bob. The traffic traverses the anonymous network and arrives at the client-side SDN switch. The client-side SDN switch can demodulate the traffic and discover a signal based on the demodulation. If the discovered signal is the same or similar to the original one, we can confirm the communication relationship between Alice and Bob.

B. Step 1: Generating the signal

We inspect traffic towards Bob using the SDN controller and intercept it as our target traffic. The controller at the server-side SDN switch can observe the flow information in the flow table of the SDN switch. Once the controller discovers a flow entry in the flow table including the IP address of the known server Bob, the controller sends a new flow entry to the SDN switch. The new flow entry is used to forward the target TCP traffic transmitted towards the server to the controller.

Once we discover the target traffic, we intend to generate an original signal for the traffic. We use the repetition error correcting codes to send the same original signal bits multiple times so as to add redundancy into the original signal and ensure the reliable signal delivery. We randomly generate the original signal (i.e., a sequence of binary bits) as

$$S = \{S_1, \ldots, S_i, S_{i+1}, \ldots, S_n\}, (S_i \in \{0, 1\}), \quad (1)$$

where n is the total number of original signal bits. Denote the number of redundancy for each original bit as r. Then we have the total length of the repetition signal as N, where $N = r \times n$ bits. Finally, we derive the sequence of repetition bits as

$$s = \{s_1, \ldots, s_m, s_{m+1}, \ldots, s_N\}, (s_m \in \{0, 1\}), \quad (2)$$

and then modulate the signal into the traffic in Step 3. Note that all of the repetition signal bits are randomly distributed in the signal sequence s. Let L_i^j ($j < r$ and $1 < L_i^j < N$) be the index of the j^{th} redundant bit for the original bit S_i in the repetition signal sequence s. Then we obtain a set of indexes in the repetition signal sequence for the original signal S_i as

$$L_i = \{L_i^1, \ldots, L_i^j, \ldots, L_i^r\}. \quad (3)$$

C. Step 2: Modulating the traffic

We regulate the rate of traffic sent from the server Bob so as to modulate the signal into the traffic. For the synchronization purpose between the server-side and client-side SDN switches, we wait for a time offset o and then initiate our signal modulation. To regulate the traffic rate of the server, we alter the advertised window size in the TCP packets sent from the proxy server by modifying the value of the window size field of the TCP header. According to the TCP sliding window protocol, the sender (i.e., the server) varies the send window size in light of the advertised window size of the receiver (i.e., the proxy server). In this way, we can vary the traffic rate of the server in a time interval to modulate a signal bit into the traffic. The time interval T_I is divided into a pair of equal subintervals, i.e., $< T_{I,1}, T_{I,2} >$. Specifically, to modulate the "1" bit of the signal, we reduce the traffic rate of the server by decreasing the value of the advertised window size of the proxy server in the first subinterval $T_{I,1}$. Then, we use the original advertised window size of the proxy server in the second subinterval $T_{I,2}$. As a result, the server increases the send window size and raises the traffic rate during $T_{I,2}$. To modulate the "0" bits of the signal, we keep the original window size in the first subinterval $T_{I,1}$ and reduce the window size in the second subinterval $T_{I,2}$. Thereforem the traffic rate sent by the server decreases in $T_{I,2}$. Finally, we can modulate a sequence of binary repetition bits (e.g. "01010") into the target traffic by varying the advertised TCP window size of the proxy server. We have all of the repetition bits modulated in a sequence of time intervals as

$$\begin{aligned} T = \{&< T_{I,1}(1), T_{I,2}(1) >, \ldots, < T_{I,1}(m), T_{I,2}(m) >, \\ &\ldots, < T_{I,1}(N), T_{I,2}(N) >\}, \end{aligned} \quad (4)$$

where the interval $< T_{I,1}(m), T_{I,2}(m) >$ corresponds to the m^{th} repetition bit, i.e., s_m.

D. Step 3: Demodulating the traffic

We demodulate the traffic at the client-side SDN switch so as to recover the modulated signal. The target traffic that carries the original signal passes through the anonymous network and reaches the client-side SDN switch. The client-side switch can deploy a flow entry on the SDN switch so as to forward the traffic transmitted towards the client Alice to the controller. After receiving the target traffic, the controller can record the timestamp and size of each packet towards the client Alice. Then the traffic is sent back to the server-side SDN switch and transmitted to the Alice. Recall that the signals at the server-side switch are modulated into the traffic after a time offset. After a one-way delay between Alice and Bob, the traffic reaches the client-side SDN switch through the anonymous network. Therefore, to correctly demodulate the traffic, we should start at a time offset equal to the one-way delay. However, we cannot accurately predict the one-way delay between Alice and Bob due to the dynamic nature and complexity of the Internet and anonymous proxy servers. Denote the worst case one-way delay between the client and the server as D. Then the traffic carrying the signal can arrive at the client during the period $(o, o + D]$. We use a sliding window W_i to accurately determine the time offset for demodulating a signal. The sliding window should be carefully selected. Smaller the sliding window is, more accurate the found start time of the signal is. Therefore, we skip the traffic heuristically by using different time offsets $o + W_i * q$ where q

is the step of the sliding window and stop the skipping as long as the signal is detected. We discuss the practical selection of the sliding window and its step in Section V in detail. The traffic starting at $o + W_i * q$ is divided into segments using half of the time interval $T_I/2$. A traffic rate time series can be obtained by calculating the traffic rate in each time interval $T_I/2$. Denote a pair of traffic rate in the m^{th} time interval $T_I(m)$ as $< x_1(m), x_2(m) >$, where $x_1(m)$ and $x_2(m)$ are the traffic rates in the first and the second subinterval, i.e., $T_{I,1}(m)$ and $T_{I,2}(m)$, respectively. Then we can obtain the time series of traffic rate by

$$X(T_I) = \{< x_1(1), x_2(1) >, \ldots, < x_1(m), x_2(m) >, \ldots, < x_1(N), x_2(N) >\}. \quad (5)$$

To determine the signal demodulated from the traffic, we use a decision rule as

$$s'_m = \begin{cases} 1, & x_1(m) < x_2(m) \\ 0, & x_1(m) > x_2(m) \end{cases}. \quad (6)$$

Finally, we can derive a series of demodulated repetition bits as

$$s' = \{s'_1, \ldots, s'_m, s'_{m+1}, \ldots, s'_N\}, (s'_m \in \{0, 1\}). \quad (7)$$

We determine if the above discovered signal is modulated by ourselves in Step 4.

E. Step 4: Recognizing the signal

Once we derive the demodulated repetition signal, we enters the phase of recognizing the original signal. Since we have the indexes of the repetition bits for the original signal, the original signal can be recovered by accumulating the demodulated repetition bits as

$$S'_i = \begin{cases} 1, & \frac{1}{r}\sum_{m \in L_i} s'_m > 0.5 \\ 0, & \frac{1}{r}\sum_{m \in L_i} s'_m < 0.5 \end{cases}. \quad (8)$$

Then we can have the recovered signal $S' = \{S'_1, \ldots, , S'_n\}$. To determine if the recovered signal is embedded by ourselves, we compare the recovered signal S' with the original one S by using the Hamming distance $H(S', S)$. If the Hamming distance between these two signals is smaller than a threshold h ($0 \leqslant h < n$), we can determine that the original signal is recognized and the communication relationship between Alice and Bob can be confirmed.

IV. Analysis

In this section, we first analyze the regulatable range of TCP window size and then define the performance metrics for measuring the feasibility and effectiveness of our SND based traceback technique.

A. Selection of TCP Window Size

We set an appropriate advertised window size of TCP packets from the receiver (i.e., the proxy server) at the server-side switch in order to change the TCP transmission rate of the sender (i.e., the server Bob). Since the selective-repeat sliding window protocol is pervasively used in various modern operating systems, we take this protocol as an example to perform the theoretical analysis in this paper. The send window (*SWND*) is used to control the amount of data transmitted under the limit of the minimum of the sender's congestion window (*CWND*) and the receiver's advertised window (*AWND*). Denote SWND, AWND, and CWND as W_s, W_a, and W_c, respectively. Then we can have

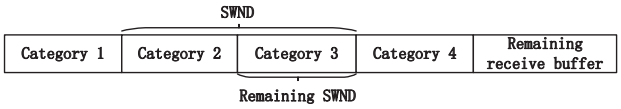

Fig. 2. 4 categories of data in the TCP buffer of the sender

$$W_s = min(W_c, W_a), \quad (9)$$

where CWND controlled by the TCP slow start algorithm is used for the sender's flow control to avoid overloading network traffic, and AWND is used by the receiver to inform the sender how much data can be sent so as to avoid overloading receiver buffer. The CWND size is multiples maximum segment size (MSS). The default maximum segment size (MSS) is 1460 bytes. According to the slow start algorithm, the CWND size can significantly increase in a few seconds at the beginning of the TCP connection and grow larger than that of AWND. At this point, CWND has no effect on the transmission of data at the sender side and the size of SWND relies on that of AWND. Since the size of AWND is set in the window size field of the TCP header by the receiver, we can intercept the TCP packets from the receiver on the server-side SDN controller and modify the value of AWND so as to control the SWND.

By reverse-engineering the TCP protocol stack, we discover that the range of AWND depends on the remaining size of the receive buffer at the receiver side. The current AWND size at the receiver side is calculated in terms of the total size and the remaining size of the TCP receive buffer at the receiver side. By reverse-engineering the kernel source code, we summarize the method as shown in Algorithm 1. According to the algorithm, we can have the range of AWND by

$$min\{\frac{3}{4}\gamma, R_s\} - \text{MSS} < W_a < min\{\frac{3}{4}\gamma, R_s\}, \quad (10)$$

where γ is the remaining size of the receive buffer and R_s is the threshold of the AWND size. R_s is dynamically adjusted to control the growth of the AWND size. When the amount of the received data is less than half size of the receive buffer, it can be set by

$$R_s \leqslant min\{\alpha, \frac{3}{4}\beta\}, \quad (11)$$

where α is the maximum AWND size that can be advertised and β is the maximum receive buffer size. The default value of α and β is 32,767 B and 512 KB, respectively. When the amount of the received data exceeds the half size of the receive buffer, the maximum of R_s is 5,840 bytes.

In addition to the remaining size of the receive buffer at the receiver side, the remaining size of the SWND at the sender side can affect the range of AWND. As shown in Figure 2, the data in the TCP buffer of the sender can be classified into 4 categories: 1) the data is sent and acknowledged. 2) the data is sent but not acknowledged yet. 3) the data is not sent and the receiver is ready to receive it. 4) the data is not sent and the receiver is not ready to receive it. The size of SWND covers the sizes of both the 2nd and the 3rd category data. Thus, the remaining size of the SWND is the size of the 3rd category data that can be sent by the sender. Denote the size of the 3rd category data as W_u. The sender chooses the maximum of the AWND and W_u as the current SWND size to determine the

size of the data that can be sent. Therefore, the size of AWND should be larger than the remaining size of the SWND as

$$W_u \leqslant W_a. \tag{12}$$

Algorithm 1 Calculating a new advertised window size

Require:
(a) δ: the total receive buffer size,
(b) θ: three quarters of the remaining receive buffer size,
(c) ε: a window scaling.
Ensure: a new advertised window size W_n

```
δ = min{α, (3/4)β}
θ = (3/4)γ
if MSS > δ then
    MSS ← δ
end if
if receiving buffer is half full then
    if memory has pressure then
        limit R_s under 5840 bytes
    end if
    if θ < MSS then
        return 0
    end if
end if
if θ > R_s then
    θ ← R_s
end if
W_n ← W_a
if ε ≠ 0 then
    W_n ← θ
else if |W_a − θ|==MSS then
    W_n ← (θ/MSS)×MSS
else if θ==MSS and θ > W_a + (δ >> 1) then
    W_n ← θ
end if
return W_n
```

According to Equation (10), (11), and (12), we can derive the range of AWND by

$$max\{min\{\frac{3}{4}\gamma, R_s\} - \text{MSS}, W_u\} < W_a < min\{\frac{3}{4}\gamma, R_s\}. \tag{13}$$

In practice, the first regulated AWND should be smaller than the latest unregulated AWND so as to reduce the size of SWND and the sender's transmission rate. Therefore, we can obtain

$$W_a < W'_a, \tag{14}$$

where W'_a is the latest unregulated AWND. Finally, we can refine the range of AWND by

$$max\{min\{\frac{3}{4}\gamma, R_s\} - \text{MSS}, W_u\} < W_a < min\{\frac{3}{4}\gamma, R_s, W'_a\}. \tag{15}$$

B. Performance Metrics

To validate the detection of the signal modulated into the target traffic, we leverage two metrics, i.e., detection rate and false positive rate. P_d is the probability that an original signal bit is correctly recognized. Recall that there are r (repetition) bits in one original signal bit. If $\lfloor r/2 \rfloor + 1$ repetition bits are correctly recognized, the original signal can be identified. Denote the number of correctly recognized repetition bits as Y. We can obtain

$$\begin{aligned} P_d &= P(Y \geqslant \lfloor r/2 \rfloor + 1) \\ &= 1 - P(Y < \lfloor r/2 \rfloor + 1) \\ &= 1 - \sum_{i=0}^{\lfloor r/2 \rfloor} \mathcal{C}_r^i P_r^i (1 - P_r)^{r-i}, \end{aligned} \tag{16}$$

where P_r is the probability that one repetition signal bit is correctly recognized. Since P_d is a monotonously increasing function with respect to the number of redundancy r, we can raise the detection rate by increasing r.

The detection rate $P_{D,n,h}$ is defined as the probability that the number of the unrecognized original signal bits cannot exceed the threshold h of Hamming distance. Denote the number of unrecognized original signal bits as Z. Given P_d for 1-bit original signal, we can derive

$$\begin{aligned} P_{D,n,h} &= P(Z \leqslant h) \\ &= \sum_{i=o}^{h} \mathcal{C}_n^i P_d^{n-i} (1 - P_d)^i. \end{aligned} \tag{17}$$

The false positive rate $P_{F,n,h}$ is the probability that a signal is found in unmodulated traffic. The traffic that does not carry the signal is referred to as the clean traffic. Denote the probability that the original signal bit 0 is detected in the clean traffic as $P_{d,0}$ and the probability that the original signal bit 1 is detected in the clean traffic as $P_{d,1}$. Then the false positive rate can be computed by

$$\begin{aligned} P_{F,n,h} &= P(Z \leqslant h) \\ &= \sum_{i=0}^{h} \mathcal{C}_n^i (\frac{P_{d,0} + P_{d,1}}{2})^{n-i} (1 - \frac{P_{d,0} + P_{d,1}}{2})^i, \end{aligned} \tag{18}$$

where n is the original signal length. As we can see from this equation, we can effectively decrease the false positive rate by raising the original signal length. However, the false positive rate can grow by raising the Hamming distance threshold h.

V. Experimental Evaluation

We implement the SDN-based traceback system in the real-world network environment. In this section, we evaluate the feasibility and effectiveness of our technique using three different anonymous communication systems including SSH, OpenVPN, and Tor. All the experiments are performed in a controlled manner. To avoid legal issues, the TCP traffic used for experiments is generated by our ourselves.

A. Experimental Setup

In our experimental setting, we deploy a client, a web server, a remote proxy server, two SDN switches and two SDN controllers. At the web server side, an Apache web service is installed and a file is put on the web server located on our campus. To evaluate the effectiveness of our trackback technique for different anonymous systems, we install two types of single-hop anonymous systems (i.e., SSH and OpenVPN) and a multi-hop anonymous system (i.e., Tor) at the client side. In addition, the setting of the socks5 proxy is configured in a Firefox browser as the SSH and Tor client provides the socks5 local proxy. For the single-hop anonymous systems,

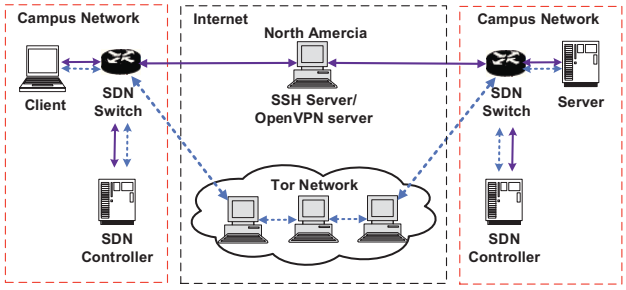

Fig. 3. Experiment setup

both SSH and OpenVPN services are installed in a single-hop anonymous server located in North America. The client leverages the SSH or VPN tunnel to download the file on the server. For the multi-hop anonymous network, the Tor client first chooses three Tor relay servers around the world in terms of the Tor path selection algorithm [4] and establishes a three-hop path in the Tor network. Then we can download the file using the firefox through the Tor client.

A pair of a Pica8 SDN switch [19] and a Floodlight controller [21] is deployed at the server side and the other pair is installed at the client side. Since the setup at both the client side and the server side is the same, we take the setup at the server side as an example. We use a Pica8 P-3297 switch that contains a serial console port, $2 \times 1Gb$ management ports, $48 \times 1Gb$ ethernet ports, and $4 \times 10Gb$ SFP+ ports. The Pica8 P-3297 switch runs an open network operating system that supports Open-vSwitch (OVS) and OpenFlow protocol. We use a computer equipped with two Ethernet interfaces as the SDN controller and install Floodlight on the machine. One Ethernet interface on the computer is connected to the serial console port of the SDN switch so as to configure the switch. The other Ethernet interface on the computer is connected to a management port on the SDN switch in order to allow the controller to communicate with the SDN switch using the OpenFlow protocol. We first configure the setup of the SDN switch through the serial console port. We set the mode of the SDN switch as OVS and the version of the OpenFlow protocol as 1.3. Then we configure a bridge on the switch and add two Ethernet ports to the bridge. These two Ethernet ports in the switch are used to connect our campus network and the server. In addition, the IP address of the controller and the Floodlight service port are set to allow the OVS in the SDN switch to connect to the Floodlight service in the controller through the management port. Once the configuration is done, the controller can communicate with the SDN switch. Then a flow entry is sent to the SDN switches so as to force the switch to forward the target traffic to our controller. The same process goes through at the client side. Finally, we modify the packet message processing module of the Floodlight source code in the controller to implement the functionalities of the signal modulation at the server side and signal recognition at the client side, respectively.

B. Experimental Results

To evaluate the effectiveness of the traceback technique, the client downloads the file 50 times using SSH tunnel, OpenVPN, and Tor, respectively. At the server-side SDN switch, we generate a random signal with 24 bits. Upon completing the TCP 3-way handshake between the proxy server and the web sever, we set the time offset o as 10 seconds. After that, we initiate to modulate the signal into the traffic by varying the TCP advertised window size of the packets from the proxy server. At the client-side SDN switch, we record the timestamps and packet sizes of the rest of the TCP packets from the proxy server after the TCP 3-way handshake between the client and the proxy server. The default values of the ratio of the current AWND size W_a to the latest unregulated AWND size W'_a, the time interval T_I, redundancy r, Hamming distance threshold h, and signal length n used in the experiments are 3/4, 800 ms, 6, 7, and 24 bits, respectively. The default sliding window size for SSH and OpenVPN is 50 ms, while the default sliding window size for Tor is 200 ms. We evaluate the detection rate by varying the value of one of these variables while keeping other variables at the default values.

To validate the false positive rate, we let the client respectively downloads 50 files using SSH tunnel, OpenVPN, and Tor again. However, no signal is modulated into the traffic at the server-side SDN switch this time. Then we generate random signals and demodulate signals from the traffic at the client-side switch. By computing the number of signal bits detected in the traffic, we can derive the false positive rate.

Figure 4 illustrates the relationship between the detection rate and the sliding window size for Tor. In light of the empirical cumulative distribution function of the one-way delay over Tor [20], the longest one-way delay is around 2 seconds. Since an end-to-end anonymous communication path in the Tor network includes four TCP connections, the average longest one-way delay between two hosts in a TCP connection is around 500 microseconds in the Tor network. After determining the longest one-way delay in the Tor network, we should carefully choose the step of the sliding window to check when the traffic carrying the signal arrives at the client-side switch. As shown in Figure 4, the detection rate is the highest by using the sliding window size W_i as 200 ms. Therefore, we set the sliding window size W_i at 200ms in order to determine the time delay. The results of the detection rates for Tor are shown in Figure 5 by varying the step of the sliding window. According to the figure, when the step of sliding window reaches 5, that is, the time delay is 1000 ms, we can obtain the best detection rate. Note that, the time offset o is set at 10 seconds. It reveals that the signal can be detected after 11 seconds due to the large one-way delay for the Tor network. Moreover, no matter what the time delay is, the false positive rates are very low (less than 0.5%).

Figure 6 demonstrates the relationship between the detection rate and the sliding window size for SSH and OpenVPN. It can be observed that the detection rates for SSH and OpenVPN are almost the same given the same sliding window sizes. When we set the sliding window size W_i at 30 ms or 50 ms, the corresponding detection rates reach the peak, 100%. Therefore, we set the sliding window size W_i at 50 ms so as to determine the time delay. The results are shown in Figure 7 by varying the step of the sliding window. Recall that the average longest one-way delay between two hosts in a TCP connection is around 500 microseconds in the Tor network. The longest one-way delay of the communication through the SSH and OpenVPN can be around 1 second as there are two connections in their paths. After determining the longest one-way delay in the SSH and OpenVPN, we select the step of sliding window with which the highest detection rate can be

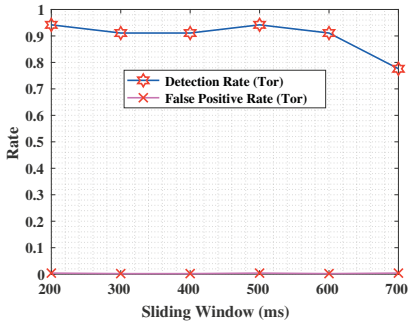

Fig. 4. Detection rate for Tor versus Sliding Window

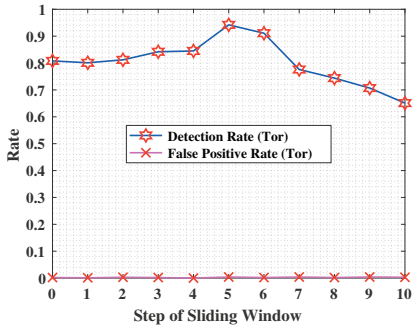

Fig. 5. Detection rate for Tor versus Step of Sliding Window (200 ms)

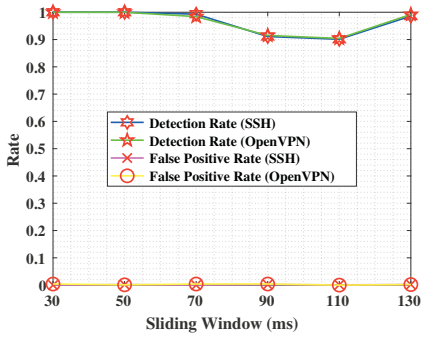

Fig. 6. Detection rate for SSH and OpenVPN versus Sliding Window

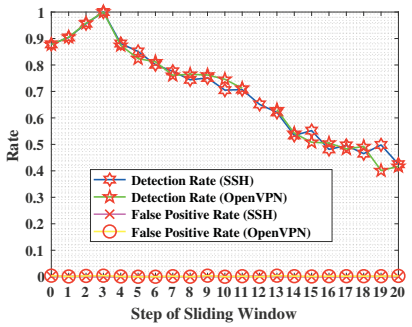

Fig. 7. Detection rate for SSH and OpenVPN versus Step of Sliding Window (50 ms)

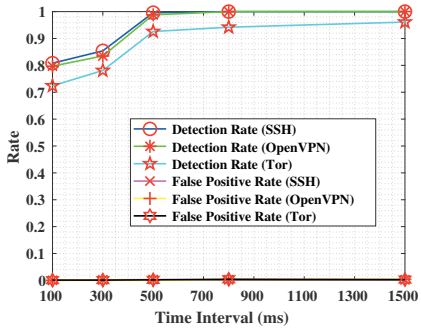

Fig. 8. Detection rate versus time interval

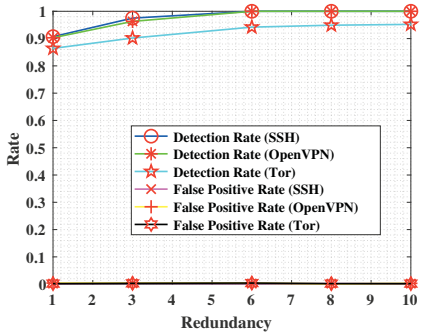

Fig. 9. Detection rate versus redundancy

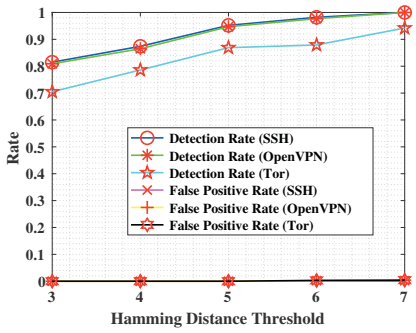

Fig. 10. Detection rate versus Hamming distance threshold

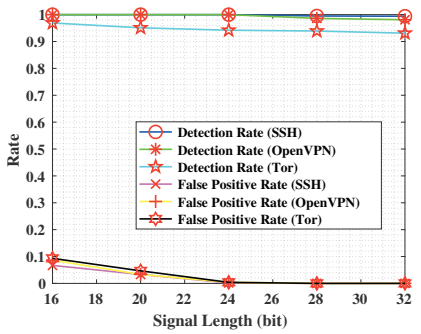

Fig. 11. Detection rate versus signal length

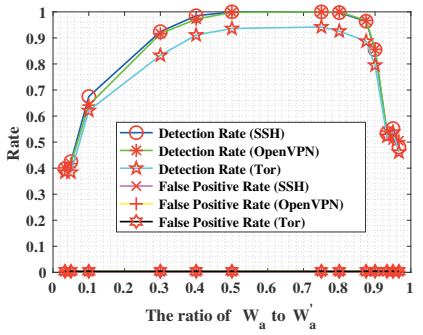

Fig. 12. Detection rate versus the ration of W_a to W'_a

obtained. As shown in Figure 7, the detection rate for SSH and OpenVPN in our experiments can approach 100% when setting the step of sliding window at 3. Accordingly, the optimal time delay is 150 ms. Note that, the time offset o is set at 10 seconds. Therefore, the traffic carrying the signal arrives at the client-side switch after 10.15 seconds. Since the optimal time delay is 150 ms, when the sliding window size is 130 ms, the detection rate is just a little bit lower than 100% as shown in Figure 6. Moreover, no matter what the time delay is, the false positive rates are very low (less than 0.5%).

Figure 8 gives the detection rate in light of time interval used for modulating the signal. As shown in this figure, the detection rate can significantly rise when the time interval slightly increases. The detection rates for both SSH and OpenVPN are almost the same given the same time intervals. When the time interval is 800 ms, the detection rates for both SSH and OpenVPN can approach 100%. However, the detection rates for Tor are comparatively lower given the same time intervals since the Tor traffic rate is unstable as Tor uses a three-hop path to transmit the data between the client and the server. Consequently, the detection rate for Tor is around 94.2% using the time interval 800 ms and it can reach 96.7% using the time interval 1500 ms. Moreover, as we can see from this figure, the false positive rates are fairly low (less than 0.5%) and slightly decrease by raising the time interval.

Figure 9 illustrates the correlation between the detection rate and redundancy. As illustrated in the figure, the detection rates grow by increasing the number of redundant signal bits. The detection rate can approach 100% when we use 6 redundant signal bits to encode one original signal. Furthermore, the false positive rates for the three distinct anonymous network systems are almost the same by varying the number of redundancy.

Figure 10 shows the relationship between detection rate and Hamming distance threshold. By using larger Hamming distance threshold, the detection rate can significantly rise.

When we use Hamming distance threshold of 7, the detection rates for SSH and OpenVPN can approach 100%. However, due to the unstable network performance of Tor, the detection rate for Tor is 94.2% using Hamming distance threshold of 7. Furthermore, the false positive rates for the three anonymous network systems are very low (less than 0.5%) although they slightly grow by increasing the Hamming distance threshold.

Figure 11 depicts the correlation between the detection rate and signal length. As shown in the figure, the detection rates for the three anonymous network systems are almost the same given the same signal lengths. However, the false positive rates are considerably decreased. The false positive rates are below 0.5% using a 24-bit signal, while the false positive rates approach zero using a 32-bit signal. This matches our theoretical analysis in Section IV-B. Since it requires more time to modulate a longer signal, we use the 24-bit signal as the default signal length in our experiments.

Figure 12 illustrates the relationship between the detection rate and the ratio of the current AWND size W_a to the latest original AWND size W'_a. We vary the ratio of W_a to W'_a to empirically evaluate the maximum and minimum values of regulatable advertised TCP window size in Equation (15). As we can see from the figure, when the range of the ratio of W_a to W'_a is between 0.5 and 0.8, the detection rates are optimal (100% for SSH and OpenVPN and 94.2% for Tor) and the false positive rates are fairly low (less than 0.5% for the three anonymous network systems). It matches our theoretical analysis in Section IV-A. However, smaller the ratio is, lower the traffic rate is. To avoid considerably reducing the traffic rate and keep our traceback more stealthy, we use 3/4 as the ratio of W_a to W'_a.

VI. Related Work

We are the first to vary the advertised TCP window size and embed a signal into the traffic to confirm the communication relationship between a user and a server. Since we just modify the TCP window size on the SDN controller, this technique can be easily implemented and deployed over SDN. Extensive real-world experimental results demonstrate the feasibility and effectiveness of this technique.

End-to-end Traffic analysis technique is a major means to trace the communication between the sender and the receiver who are communicating with each other using anonymous communication systems. The traffic analysis technique can be classified into two categories: passive traffic analysis [9], [31] and active watermarking techniques [10], [13], [18], [27]–[29]. To perform the passive traffic analysis, sniffers should be deployed at both the client side and the server side so as to passively record the inbound and outbound traffic of the client and server. By comparing the similarity of the patterns of the traffic between the client and the server, the operator can determine their communication relationship. However, to enhance the detection rate and reduce false positive rate, the operator should monitor the traffic for a reasonably long time. The active watermarking technologies are to actively manipulate various characteristics of the network traffic so as to embed a secret watermark into the traffic at the server/client side. Then the traffic carrying the watermark traverses the anonymous communication system and arrives at the client/server side. If the watermark can be identified, the communication relationship between the client and server can be confirmed. Such techniques can significantly reduce the false positive rate using a sufficiently long signal, and they do not require the very long training on the traffic, which is generally required in passive traffic analysis.

Different features of network traffic can be used as watermark carriers. For example, Ramsbrock *et al.* [23] varies the length of packets at the application layer and embedded the watermark by inserting characters in the packet. Since the content of the packets is modified, the embedded watermark could be easily discovered by the user or the server. Ling et al. [10] proposes packet size-based covert channel attacks against a single-hop anonymous communication system, i.e., Anonymizer. The attacker changes the size of packets transmitted between the malicious website and the anonymous server so as to embed the watermark into the traffic. Another attacker sniffs the traffic and recognizes the signal to confirm the communication relationship between the malicious website and the client. Yu *et al.* [29] develops an invisible traceable watermarking technique based on Direct Sequence Spread Spectrum (DSSS) using pseudo-noise (PN) codes. Jia *et al.* [7] proposes a single-stream scheme to detect malicious DSSS watermark blindly.

The packet timing information is also used as a watermark carrier. Wang *et al.* [28] propose a watermarking technique based on inter-packet delay (IPD). The basic idea of this technique is introducing a watermark by slightly adjusting the IPD of the selected packets. Based on the analysis of [28], Peng *et al.* [18] propose an IPD-based watermarking detection approach using adjacent intermediate agents. Experimental results show that this countermeasure scheme can effectively identify the watermark. Moreover, when the parameters used for encoding the watermarking are not carefully selected, the entire watermarking can be correctly recovered. To reduce the impact on network delay disruption, Wang *et al.* [27] propose a watermarking technique for tracking P2P anonymous VoIP phones over the Internet. Experimental studies shown that this technique can effectively be applied to any P2P anonymous VoIP phone that lasts longer than 90 seconds.

Houmansadr *et al.* [2] design the RAINBOW watermarking scheme, which uses a non-blind watermarking approach to manipulate IPD and use spread-spectrum techniques to reduce latency. June *et al.* [22] and Wang *et al.* [26] propose two watermarking schemes based on the interval of time respectively. The two schemes adjust the number of packets falling in the interval or change the arrival time of packets in the interval in order to embed the watermark. Such schemes can effectively reduce the impact of network flow conversion problems, and they have good robustness, but require more packets.

To combat multi-stream attacks, Houmansadr *et al.* [5] design the SWIRL watermarking scheme. It was the first blind watermarking that can be used for large-scale traffic analysis. It uses a stream-dependent approach to resist multi-stream attacks and Tor network congestion and mark each stream in different modes. SWIRL is robust to loss of packet and network jitter and it is invisible to users and attackers.

These watermarking technologies can exist under confrontational network conditions and can evade the existing detection methods. Therefore, Luo *et al.* [14] propose a new detection system named BACKLIT, based on the change of inherent timing features of TCP streams to detect the watermark. The program had been tested on the PlanetLab and the results show that BACKLIT can detect the most advanced

watermarking schemes based the arriving time of packets with high accuracy and low error rates.

Research efforts also presented how to trace the Tor hidden service [3], [11], [15], [17], [30]. For example, Ling *et al.* propose a protocol-level Tor hidden server traceback technique. The attacker actively forces the hidden server to produce a protocol feature and tries to discover the feature at the Tor entry node. Once the feature is confirmed, the IP address of the hidden server can be identified. Tian *et al.* [25] investigate how to perform traceback over the Freenet.

VII. Conclusion

In this paper, to address the increasingly serious abuse issues of anonymous communication systems, we introduce a novel and practical SDN-based traceback technique to confirm the communication relationship between users and servers. At the server-side SDN switch, we leverage the SDN controller to intercept the traffic and change the advertised TCP window size of the packets which are transmitted towards the target server. In this way, we can modulate a secret signal into the traffic by varying its traffic rate at the server. Repetition error correcting codes are applied to enhance the robustness of the modulated signal. Furthermore, based on the comprehensive theoretical analyses, we perform extensive empirical experiments to discover the regulatable range of the advertised TCP window size so as to exert a minimal impact on the traffic rate. Finally, extensive real-world experiments are conducted using three anonymous communication systems, i.e., SSH, OpenVPN, and Tor to verify the feasibility and effectiveness of our traceback technique.

Acknowledgments

This work was supported in part by National Key R&D Program of China 2018YFB0803400 and 2017YFB1003000, National Science Foundation of China under Grant 61572130, 61532013, and 6163200, by the US National Science Foundation under Grant Nos. (1642124 and 1547428), Jiangsu Provincial Key Laboratory of Network and Information Security under Grant BM2003201, Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grant 93K-9, and Collaborative Innovation Center of Novel Software Technology and Industrialization. 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 funding agencies.

References

[1] 94% of Tor Traffic is Malicious, According To CloudFlare. https://darkwebnews.com/anonymity-tools/tor/tor-malicious-cloudflare/, 2018.

[2] N. B. Amir Houmansadr, Negar Kiyavash. 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.

[3] A. Biryukov, I. Pustogarov, and R.-P. Weinmann. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P)*, 2013.

[4] R. Dingledine and N. Mathewson. Tor Path Specification. https://gitweb.torproject.org/torspec.git/tree/path-spec.txt, 2018.

[5] A. Houmansadr and N. Borisov. SWIRL: A Scalable Watermark to Detect Correlated Network Flows. In *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS)*, 2011.

[6] A. Houmansadr, N. Kiyavash, and N. Borisov. Non-Blind Watermarking of Network Flows. 2014.

[7] W. Jia, F. TSO, Z. Ling, X. Fu, D. Xuan, and W. Yu. Blind detection of spread spectrum flow watermarks. In *Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM)*, Rio de Janeiro, Brazil, April 2009.

[8] B. N. Levine, M. Liberatore, B. Lynn, and M. Wright. Statistical Detection of Downloaders in Freenet. In *Proceedings Third IEEE International Workshop on Privacy Engineering*, pages 25–32, May 2017.

[9] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright. Timing Attacks in Low-Latency MIX Systems. In *Proceedings of Financial Cryptography (FC)*, February 2004.

[10] Z. Ling, X. Fu, W. Jia, W. Yu, and D. Xuan. Novel Packet Size Based Covert Channel Attacks against Anonymizer. 2013.

[11] Z. Ling, J. Luo, K. Wu, and X. Fu. Protocol-level Hidden Server Discovery. In *Proceedings of the 32th IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[12] Z. Ling, J. Luo, K. Wu, W. Yu, and X. Fu. TorWard: Discovery of Malicious Traffic over Tor. In *Proceedings of the 33rd IEEE International Conference on Computer Communications (INFOCOM)*, 2014.

[13] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia. A New Cell Counting Based Attack Against Tor. In *Proceedings of ACM CCS*, November 2009.

[14] X. Luo, P. Zhou, J. Zhang, R. Perdisci, W. Lee, and R. K. Chang. Exposing Invisible Timing-Based Traffic Watermarks with BACKLIT. In *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC)*, 2011.

[15] S. J. Murdoch. Hot or Not: Revealing Hidden Services by Their Clock Skew. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, November 2006.

[16] T. OConnor, W. Enck, W. M. Petullo, and A. Verma. PivotWall: SDN-Based Information Flow Control. In *Proceedings of the ACM Symposium on SDN Research (SOSR)*, 2018.

[17] L. Øverlier and P. Syverson. Locating Hidden Servers. In *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, 2006.

[18] P. Peng, P. Ning, and D. S. Reeves. On the Secrecy of Timing-based Active Watermarking Trace-back Techniques. In *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.

[19] Pica8 Inc. Pica8 C Open Enterprise Networking Solutions. https://www.pica8.com/, 2018.

[20] R. Pries, W. Yu, S. Graham, and X. Fu. On Performance Bottleneck of Anonymous Communication Networks. In *Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, April 14-28 2008.

[21] Project Floodlight. Floodlight OpenFlow Controller. http://www.projectfloodlight.org/, 2018.

[22] 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 INFOCOM*, May 2007.

[23] D. Ramsbrock, X. Wang, and X. Jiang. A First Step Towards Live Botmaster Traceback. In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2008.

[24] The Tor Project, Inc. Tor: Anonymity Online. https://www.torproject.org/, 2018.

[25] G. Tian, Z. Duan, T. Baumeister, and Y. Dong. Traceback Attacks on Freenet. 2017.

[26] 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.

[27] X. Wang, S. Chen, and S. Jajodia. Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet. In *Proceedings of ACM CCS*, November 2005.

[28] X. Wang and D. S. Reeves. Robust Correlation of Encrypted Attack Traffic Through Stepping Stones by Manipulation of Inter-packet Delays. In *Proceedings of ACM CCS*, November 2003.

[29] 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)*, 2007 May.

[30] L. Zhang, J. Luo, M. Yang, and G. He. Application-level attack against Tor's hidden service. In *Proceedings of the 6th International Conference on Pervasive Computing and Applications*, 2011.

[31] 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.