Self-Adaptive Anonymous Communication Scheme Under SDN Architecture

Tingting Zeng*, Meng Shen*†, Mingzhong Wang‡, Liehuang Zhu*, Fan Li*
*Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications,
School of Computer Science, Beijing Institute of Technology, P. R. China
†Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, P. R. China
‡Faculty of Arts and Business, University of the Sunshine Coast, Queensland, Australia
Email: {wilmaztt, shenmeng, liehuangz, fli}@bit.edu.cn; mwang@usc.edu.au

***Abstract*—Communication privacy and latency perceived by users have become great concerns for delay-sensitive Internet services. Existing anonymous communication systems either provide high anonymity at an expense of prolonged latency (e.g., mix-net), or offer better real-time performance by sacrificing the ability against traffic analysis attacks (e.g., Onion Routing). The emerging Software-Defined Networking (SDN) introduces additional challenges to communication anonymity, due to the existence of a centralized controller that has a global view of the entire network traffic.**

In this paper, we propose a new anonymous communication scheme for delay-sensitive services under SDN scenarios, which can simultaneously protect communication privacy and reduce the end-to-end latency. A self-adaptive method based on the mix-net framework is designed to dynamically modify the waiting threshold of mix nodes, which helps to reduce the communication latency. In order to preserve the degree of anonymity, the self-adaptive method is incorporated with a random walking strategy for packets forwarding. Both theoretical analysis and experimental results prove that our scheme provides a moderate degree of anonymity and effectively reduces the latency derived from mix-net by up to 50%.

***Index Terms*—Privacy Preservation; Self-Adaptive; Anonymous Communication; Software-Defined Networking (SDN)**

I. Introduction

The diversification of network services and applications has imposed great challenges on the traditional rigid networking architectures. As an emerging approach to meet these challenges, Software-Defined Networking (SDN) appears to be more flexible, dynamic and future-proofing. Among the leading reference implementations of SDN, OpenFlow [2, 3] defines a standard for communications between control and data planes. In the OpenFlow protocol, a centralized controller, which has a global view of the entire network, is employed to calculate appropriate routes for each flow. Functions of switches degenerate to only deal with data packets passively according to forwarding rules made by the controller. The tremendous power of controllers raises privacy concerns when user traffic traverses the network based on OpenFlow. If network operators are curious about the interactions between endpoints, they could easily perform traffic analysis on records collected at the controller.

In traditional networking scenarios, anonymous communication has attracted much research attention. A variety of anonymous communication systems have been proposed to hide the relationship between a user and its communication correspondences. The mix-net is referred to as the first anonymous communication system [4]. The basic idea of mix-net is to batch relaying messages through a path of "mixes" by wrapping them in layers of public-key cryptography, and hide the correspondence between the input and output of each mix by delaying and re-ordering. Although anonymity is well protected, it introduces relatively high latency. Therefore, this scheme can only be applied to delay-tolerant applications, such as e-mails.

In order to improve communication efficiency, anonymous communication systems with low latency are proposed. The most popular one is Tor [8], which is derived from onion routing. In Tor, each onion router knows only its predecessor and successor. Therefore, any individual router cannot infer the ultimate source or destination of a message. However, Chakravarty et al [12] put forward a method that can reveal the actual sources of anonymous traffic. This approach is based on correlation analysis of network traffic and achieves 81.4% accuracy in real-world experiments. These findings raise an increasing number of privacy concerns, because the assumption of controlling a particular server in [12] is no longer necessary in SDN scenarios.

Based on the discussions above, it is clear that for latency-sensitive applications (e.g., web browsing), achieving anonymity and real-time performance simultaneously remains a great challenge in SDN scenarios. In this paper, we present the design of an anonymous communication scheme, which can resist global traffic analysis attacks and provide low communication latency.

Inspired by the mix-net system, we also employ the mix strategy at intermediate nodes to provide high anonymity. The end-to-end latency in mix-net is mainly caused by the traffic-agnostic forwarding policy at mix nodes, i.e., each mix node keeps the received packets for a *fixed* amount of time before sending them out, unaware of the traffic load. Based on this observation, we introduce a self-adaptive method that dynamically modifies the waiting threshold of mix nodes according to varying traffic loads.

Since the source routing strategy is adopted in both mix-net and Tor systems, the self-adaptive method may reduce the confusion caused by mixing operations and thereby increase the risk of traffic analysis attacks. To address this challenge, we employ the random walking strategy, instead of the source routing, to avoid the communication relationships being deanonymized when forwarding reply messages back to the source. The self-adaptive method incorporated with the random walking strategy can achieve the goals of our design.

To evaluate the effectiveness of the proposed scheme, we verify the degree of anonymity on the basis of Shannon entropy and theoretically analyze its anonymity against node collusion. In addition, we implement the scheme in Mininet and conduct extensive simulations on the impact of the self-adaptive strategy. Experimental results demonstrate that our scheme can provide high degree of anonymity and reduces the communication latency.

The main contributions of this paper can be summarized as follows:

- We propose a new anonymous communication scheme for delay-sensitive services under SDN scenarios, which ensures both anonymity and real-time performance.
- We put forward a self-adaptive method to dynamically modify the packet forwarding threshold at mix nodes, according to real-time network loads and user preferences.
- We theoretically analyze the degree of anonymity in our scheme, which shows our superiority in anonymity compared with the classic Crowds system.
- We conduct experiments using real user traffic data to show the efficiency of our scheme, which reduces the latency by at most 50%.

The remainder of this paper is organized as follows. Section 2 summarizes the related works in the fields of anonymous communication systems. Then we elaborate our anonymous communication scheme in Section 3. The theoretical and experimental evaluations of our scheme are presented in Sections 4 and 5. Finally, we conclude this paper in Section 6.

II. Background and Related work

In traditional networking scenarios, the field of anonymous communication has attracted a considerable amount of research attention. Motivated by the need for anonymity in network communications, a variety of anonymous communication systems have been proposed, which can be roughly classified into two categories. The first category is designed for high-latency anonymity networks, including mix-based systems like Mix-nets [4], Babel [19], Mixmaster [21] and Mixminion [20], which tries to increase the anonymity at the cost of prolonging communication latency. The second category is proposed for latency-sensitive scenarios. Typical designs include Onion Routing [7, 8, 13] and P2P-based schemes (e.g., DC-nets [6], Crowds [5], MorphMix [22] and Tarzan [9]). Compared to the first category, these systems can provide better real-time performance, but are more fragile in face of traffic analysis attacks.

Mix-net. Proposed in 1981, Mix-net is the first anonymous communication system [4]. The basic idea of Mix-net is to batch relaying messages through a path of "mixes" by wrapping them in layers and delaying for hours and reordering in each mix. Although the anonymity is securely protected, it introduces relatively high latency. Therefore, this scheme can only be applied to delay-tolerant applications, such as e-mails.

Onion Routing. The most popular communication system with low latency is the second-generation Onion Routing, i.e., Tor [8], improved by Dingledine. Tor is the most widely used volunteer-run network distributed across the world [14]. This scheme is based on circuit established by a number of proxy nodes called Onion Routers (ORs) and each OR knows only the predecessor and successor in a circuit. Therefore, any individual OR cannot infer the ultimate source or destination of a message.

Although Tor introduces few artificial delays to guarantee the real-time performance, the approach is obviously vulnerable to traffic analysis attacks. After a number of previous works, i.e., [11, 12, 14, 17], Chakravarty et al [12] proposed a new method to reveal the actual source of anonymous traffic by performing a statical correlation analysis on Netflow data, which can be collected by most Cisco routers with traffic monitoring functionalities. This approach achieves 100% accuracy in lab and 81.4% in real-world experiments. It is worth mentioning that, this approach can be implemented more easily in SDN scenarios, because the assumption of controlling a particular server in [12] is no longer necessary.

Compared with previous works, we insist that the main contributions of our research is to give consideration to both the real-time performance and the ability to resist a global traffic analysis attack of delay-sensitive services under SDN scenarios.

III. System Design

In this section, we start by outlining our design goals, then have a discussion on the threat model and assumptions, and finally describe our system model in details. In addition, we also develop a method to achieve trade-offs between anonymity and real-time performance.

A. Design Goals

Our scheme seeks to frustrate attackers in their attempt to link communication partners or to link a certain user with multiple communications. The most important two requirements of the communication under SDN architecture are real-time performance and the ability to resist a powerful global traffic analysis attack. So the main goal of our scheme is to achieve a higher anonymity than the existing low-latency schemes while make a suitable and tolerable compromise to real-time performance.

B. Threat Model and Assumptions

First of all, the target application of our design is web browsing under SDN architecture. We aim to protect user's communications privacy against curious attackers. So in this work, we mainly focus on the adversary on the side of a SDN

Controller. We consider the adversary is global and powerful-enough that can take advantage of the logging information collected from OpenFlow switches to perform passive traffic analysis attacks which resembles to the method of Chakravarty et al [12]. In addition, we assume that the Controller-side adversary would not perform active attacks such as deviating from the protocol or selectively denying service to some requests, in the consideration of the fact that he is an internet service provider who need to guarantee the usability of the network.

Secondly, our scheme is based on social relations, and more specifically, on the foundation of group organized by a number of people who are in the same organization, such as a department or a school. The fact that there may be someone who are curious about others' communication privacy should be considered, but an honest member in the group can be trusted not to leak or share the information he knows.

C. System Model

Our scheme is to establish an overlay network based on mix-net and random walking. The network consists of a number of nodes as members within a certain group. Each node registers only once and is associated with a local public-private key pair. The legality of each node can be verified by the organizer in the process of registration.

Each node has three kinds of identities listed as follows.

- **Client** node, from whom the traffic originates;
- **Proxy** node, who is responsible for collecting data requested by a client node;
- **Mix** node, who stores and shuffles the received messages, and forwards them to another mix node or a proxy node.

Messages are classified into 5 types: **RELAY**, **REQUEST**, **REPLY**, **KEY** and **KEYBACK** as shown in Table I.

TABLE I: Format of Each Type of Messages

RELAY	Local ID	Rest Length	Proxy IP	Data
REQUEST	Source ID	Transaction ID	Data	
REPLY	Transaction ID	Pieces Count	Piece ID	Data
KEY	Circuit ID	KEY or $ProxyID\|\|E(g^x)$		
KEYBACK	Circuit ID	KEYBACK or $g^y\|\|H(k\|\|SourceID)$		

Before the nodes exchange transaction messages, they first choose a number of proxy nodes from all the others in the group, and negotiate a session key with each proxy node using the key establishing messages similar to Tor, one hop at a time, so that the proxy node would receive a key negotiating request but not get the identity of the clients. The proxy nodes also need to assign a Source ID to each client node, in order to recognize where a corresponding reply should be sent back. Besides the session keys negotiated with their chosen proxy nodes, each node also shares a symmetric key with others for basic communications. For convenience, we define the first type of key as P-key, and the latter one as S-key.

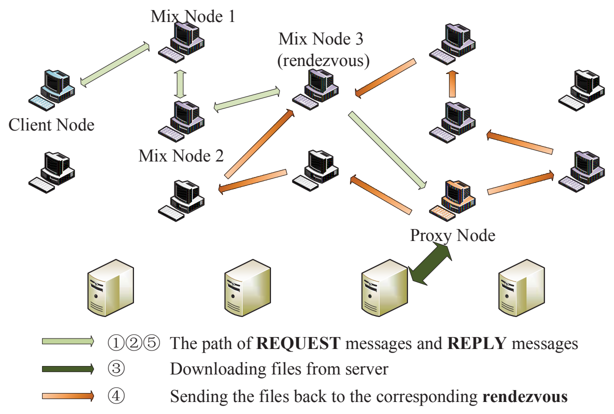

Mix Node 1
Mix Node 3 (rendezvous)
Client Node
Mix Node 2
Proxy Node
①②⑤ The path of REQUEST messages and REPLY messages
③ Downloading files from server
④ Sending the files back to the corresponding rendezvous

Fig. 1: The process of message interaction in our scheme

When a client node intents to browse a web page, it need to send a HTTP request to the corresponding web server to fetch the pages. In our scheme, as shown in Fig. 1, this process goes through five steps:

Step 1. The client node randomly selects a proxy node PN and construct a **REQUEST** message. The Source ID indicates which P-key should be used by PN, and the Transaction ID is for receiving the reply messages. The client node encrypts its HTTP request with the corresponding P-key as the data field, and wraps the **REQUEST** message in a **RELAY** message with a Local ID. Here the Local ID is a tag for recognizing which are the corresponding reply messages. Each Mix node maintains a map of $< LocalID, LastID >$. When a mix sends reply messages back to a previous node, it should look up the map, and tell its previous node which the replies belong to. The value in the rest length field is used as a time-to-live counter, which should be decreased by one before the data being forwarded to another mix node.

Step 2. Note that the client node is also a mix node. Thus, before sending out the new **RELAY**, the node needs to check whether it has received several batches of indeterminate amount of **RELAY** messages from other nodes. Every mix node should follow such a strategy, as shown in Algorithm 1. There are two parameters maintained in every node, a threshold N and a timeout T. During a period of T, denote the number of messages a mix node M_i receives and stores as n_i. If n_i is greater than or equal to N, the mix node sends all the messages to their corresponding proxy nodes. Otherwise, the mix node continues to wait. Once the timeout is triggered, the mix node either sends the stored messages directly to their corresponding proxy nodes with a probability of $\frac{n_i}{N}$, or relays the messages to another randomly selected mix node with a probability of $1 - \frac{n_i}{N}$. Any message with the Rest Length field being 0 should be discarded immediately. We refer to the last mix node before the proxy as **rendezvous**.

Step 3. When receiving a **REQUEST** message, the proxy decrypts it with the corresponding P-key, relays it to the actual destination and fetches the required web pages from the servers.

Step 4. The proxy divides the pages into pieces, which have different sizes and are labeled with a sequence of Piece IDs. Then these pieces are distributed to multiple mix nodes and relayed back to the rendezvous of the client. The forwarding strategy at each mix node is the same as described in Step 2.

Step 5. The rendezvous re-orders all the received pieces and checks the completeness of the required pages. If they are incomplete, the rendezvous requests the missed pieces; Otherwise, it sends the ordered **REPLY** messages to the client via the same path.

Algorithm 1 Forwarding Strategy of a Mix Node

```
Require: $N$ : the current value of threshold $N$
Require: $T$ : the current value of timeout $T$
Require: $Sequence$ : a sequence of messages coming into a mix node during the last period
1: while $TRUE$ do
2:     $n_i \leftarrow getLength(Sequence)$
3:     $t_c \leftarrow getCurrentTime()$
4:     if $n_i \geq N$ then
5:         Send all $S_j$ in $Sequence$ to their proxy node
6:     else
7:         if $t_c - S_0.time > T$ then
8:             if $\frac{n_i}{N} > random(0,1)$ then
9:                 Send all $S_j$ in $Sequence$ to their proxy node
10:            else
11:                Send all $S_j$ ($S_j.RestLength = 0$) in $Sequence$ to their proxy node
12:                Send all $S_j$ ($S_j.RestLength \neq 0$) in $Sequence$ to another mix node
```

D. Trade-offs Between Latency And Anonymity

Based on the above process, the dominant latency is caused by the mixing and random walking phases in Steps 2 and 4. Each relay along the random walk path might introduce delays by collecting sufficient messages to maximize the effect of confusion. Therefore, the latency is mainly determined by three factors, namely the length of the relay path L, the threshold N and the timeout T at each mix node.

The length of the relay path L is initially set by the client node in the field of Rest Length in **RELAY** message, and plays an important role in limiting the basic latency of the random walking. The other two factors are controlled by each mix node. Considering that our scheme is based on the interaction between users in the group, more active interactions will definitely lead to communications with better anonymity, however, at an expense of longer delays. In order to make our scheme more applicable to real-time sceneries, we propose a self-adaptive method to dynamically and periodically change these two parameters.

The challenge here is how to determine the density of activity of a group in a certain period of time. Once a mix node meets any condition of forwarding the messages, we refer to it as a ***saturated transmission*** if the threshold N is reached, or an ***unsaturated transmission***, otherwise. Because every next-hop node is selected randomly, a single node cannot accurately predict the time sequence of the following messages coming into it, we thus choose to take measures on rough estimation.

More specifically, we record the count of each *saturated* and *unsaturated* transmissions in a certain period of t seconds, and calculate the proportion of the *saturated* transmission to represent the fitness of the current two parameters. The fitness indicator is used to adjust the above parameters.

Algorithm 2 Calculating Parameter Values for Next Period

```
Require: $Sequence$ : a time sequence of messages coming into a mix node during the last $n$ periods
Require: $S$ : a evaluation criterion to decide the range in which to change the values
Require: $[N_{min}, N_{max}]$ : a tolerable range of threshold $N$
Require: $[T_{min}, T_{max}]$ : a tolerable range of timeout $T$
Require: $N_c$ : the current value of threshold $N$
Require: $T_c$ : the current value of timeout $T$
1: $T_{upper} = T_{max}$
2: $T_{lower} = T_{min}$
3: $R_s \leftarrow countSaturated(Sequence)$
        ▷ Calculate the proportion of saturated transmission
4: if $R_s > S$ then
5:     if $\frac{(R_s+1-2S)N}{1-S} > N_{max}$ then
6:         $N_{upper} = N_{max}$
7:     else
8:         $N_{upper} = \frac{(R_s+1-2S)N}{1-S}$
9:     $N_{lower} = N_c$
10: else
11:     if $\frac{(R_s+S)N}{2S} < N_{min}$ then
12:         $N_{lower} = N_{min}$
13:     else
14:         $N_{lower} = \frac{(R_s+S)N}{2S}$
15:     $N_{upper} = N_c$
16: $N_{bound} = (N_{lower}, N_{upper})$
17: $T_{bound} = (T_{lower}, T_{upper})$
18: for $k = 0$ through $k_{max}$ (exclusive) do
19:     $Tem \leftarrow temperature(k/k_{max})$
20:     $(N_{rand}, T_{rand}) \leftarrow pick(N_{bound}, T_{bound})$
        ▷ Pick up a pair of value randomly
21:     $dE = E(N_{rand}, T_{rand}, Sequence) - E(N_c, T_c, Sequence)$
22:     if $dE \geq 0$ or $exp(dE/Tem) > random(0,1)$ then
23:         $(N_c, T_c) = (N_{rand}, T_{rand})$
    return the final result $(N_c, T_c)$
```

Algorithm 2 depicts the process of how to calculate proper values for these parameters in the next period, based on the information collected in the last n periods.

At the beginning, all members in the group need to determine several indictors for the parameter adjustment, including a metric S, the weight of real-time performance w_r, the weight of mix performance $w_m = 1 - w_r$, and the range of N and T that they could tolerate. If the proportion of the ***saturated transmission*** R_s is larger than S, we enlarge the threshold N to at most $2N$. On the contrary, if the proportion R_s is less, N will be reduced a little to a minimum of $\frac{N}{2}$.

Then we employ another factor, i.e., timeout T. In order to balance the effect of these two parameters, we apply simulated annealing algorithm in our scheme. The search space of T is just the whole range, which has already been determined by users. Then we limit the search space of N to $[N, \frac{(R_s+1-2S)N}{1-S}]$ for $R_s > S$, or $[\frac{(R_s+S)N}{2S}, N]$ for $R_s \leq S$. Therefore, the threshold achieves its maximum value of $2N$ when $R_s = 1$ and its minimum value of $N/2$ when $R_s = 0$. The cost function of a pair of values in simulated annealing algorithm can be defined as follow:

$$E(N_{rand}, T_{rand}, Sequence) = w_r\bar{t} + w_m\frac{N_{max} - N_{rand}}{N_{max}} \tag{1}$$

where the $\bar{t}$ is the average latency in a mix node during the last n periods in the condition that the threshold is N_{rand} and the timeout is T_{rand}.

The weight of these two parts is set by users, which enables flexible adjustment between latency and anonymity. For instance, if users prefer to lower latency, they can set w_r larger than w_m. On the contrary, if they consider the mix performance to be more important, they can increase the weight of the second part. After several iterations, the Algorithm can calculate desired values of the two parameters.

IV. Theoretical Analysis

In this section, we evaluate the proposed scheme by theoretical analysis on anonymity against collaborating nodes, and make several comparisons between the effects in different conditions.

A. Evaluation Model

To quantify the anonymity against collaborating nodes provided by our scheme, we use the entropy metric [23] in the following discussion. Entropy metric considers attacker that obtains probabilistic information about users. After observing the anonymous network, the attacker assigns to the different users as being the originator of a message. The most widely used mechanism for analyzing anonymous communication is Shannon entropy, which is computed as:

$$H(X) = \sum_{i=1}^{N} -p_i log_2(p_i) \tag{2}$$

where i corresponds to each possible sender being the originator of a message in the anonymity set of size N, and p_i is their corresponding probability assigned by attackers after the attack has taken place.

We denote the entropy of the whole network by $H(X)$ and the maximum entropy by H_M. For an actual size of anonymity set, the H_M will be $log_2 N$ when all users in the anonymity set appear as being the originator with the same probability of $\frac{1}{N}$ in an ideal situation. The information an attacker benefits from the observation can be expressed by $H_M - H(X)$, and the degree of anonymity of the network can be defined as:

$$d = 1 - \frac{H_M - H(X)}{H_M} = \frac{H(X)}{H_M} \tag{3}$$

where d quantifies the amount of information leaked to the attackers ranging from 0 to 1. The combined degree of anonymity across all possibilities is calculated as

$$d = \sum_{i=1}^{K} p_i d_i \tag{4}$$

where K is the number of all possibilities, d_i is the degree obtained in different situation and p_i is the occurrence probability of each situation.

On this basis, we consider the scenario where a message arrives at a malicious mix node A_m which is collaborating with the corresponding proxy node A_p. A client chose a proxy node randomly from his trust list, while this proxy node may be a malicious node which is curious about the identity of the client. We assume that A_p organizes a group of collaborating nodes G. Let N be the number of users, and C be the number of collaborators in the collaborating group including A_p. Taking account of the fact that the nodes in G controlled by collaborators should be excluded from the anonymity set, the size of anonymity set is $N - C$, and the maximum entropy H_M should be equal to $log_2(N - C)$.

As shown in our scheme discussed above, the first collaborating mix node in the path knows only the identity of its immediate predecessor. We denote the predecessor by M, and its probability of being one of the originators of the messages by p_M. As prepared, we denote by H_k the event that the first collaborator occupies on the k-th position of the path, where the originator occupies on the 0-th position. Then $H_{k+} = H_k \vee H_{k+1} \vee \ldots \vee H_l$, where l is the maximum length of the path. Let O denote the event that the first collaborator A_m is directly preceded by the originator. On this basis, we can define the probability that the predecessor of a collaborator node is the originator as $P(O|H_{1+})$. Here $p_M = P(O|H_{1+})$.

To calculate $P(O|H_{1+})$, we first note that the probability of A_m occupying on the i-th position is

$$P(H_i) = \frac{C}{N}(\frac{N-C}{N})^{i-1}\prod_{j=0}^{i-1} P_{f\,j} \tag{5}$$

where $P_{f\,j}$ is the probability of forwarding of the j-th node on the path. Based on Eq. 5 , we can deduce that the probability of A_m occupying behind the first and second position are

$$\begin{aligned} P(H_{2+}) &= \frac{C}{N}\sum_{k=2}^{l} P(H_k) \\ &= \frac{C}{N}\sum_{k=2}^{l}\left[(\frac{N-C}{N})^{k-1}\prod_{j=0}^{k-1} P_{f\,j}\right] \end{aligned} \tag{6}$$

$$P(H_{1+}) = P(H_1) + P(H_{2+})$$
$$= \frac{C}{N} \sum_{k=1}^{l} \left[(\frac{N-C}{N})^{k-1} \prod_{j=0}^{k-1} P_{f\,j} \right] \tag{7}$$

Then based on Eq. 6 and the intuitive fact that $P(O|H_1) = 1$ and $P(O|H_{2+}) = \frac{1}{N-C}$, the probability of A_m being directly preceded by the originator $P(O)$ can be derived as follows

$$P(O) = P(H_1)P(O|H_1) + P(H_{2+})P(O|H_{2+})$$
$$= \frac{C}{N} P_{f\,0} + \frac{1}{N-C} P(H_{2+}) \tag{8}$$

Finally, we can get the probability of the predecessor of A_m being the originator :

$$p_M = P(O|H_{1+}) = \frac{P(O)P(H_{1+}|O)}{P(H_{1+})} = \frac{P(O)}{P(H_{1+})} \tag{9}$$

and the probability assigned to other honest nodes is

$$p_{m_i} = \frac{1 - p_M}{N - C - 1} \tag{10}$$

Similarly, we denote the predecessor node of A_p by D, and l_{key} the length of path in key negotiation prase. We can obtain the probability of D being the originator assigned by A_p:

$$p_D = \frac{C + 1 - q^{l_{key}-1}}{N(1 - q^{l_{key}})} \tag{11}$$

and the probability assigned to other honest nodes is

$$p_{d_i} = \frac{1 - p_D}{N - C - 1} \tag{12}$$

where $q = \frac{N-C}{N}$ represents the probability of choosing a honest node as the successor node on the key negotiation path.

Next, we have to consider the benefit obtained from both A_p and A_m. Let E denote the event that the originator is immediately preceded by at least one of A_p and A_m, and B_i denote the event that node i is the originator.

So if M and D are the same node, the probability of the each honest node i being the originator assigned by attackers can be expressed as:

$$p_i = P(B_i|E) = \begin{cases} \dfrac{p_M + p_D - p_M p_D}{\sum_{i=1}^{N-C} P(E|B_i)} & i = M \\ \dfrac{p_{m_i} + p_{d_i} - p_{m_i} p_{d_i}}{\sum_{i=1}^{N-C} P(E|B_i)} & i \neq M \end{cases}$$

If M and D are different, then the probability of each honest node i can be expressed as:

$$p_i = P(B_i|E) = \begin{cases} \dfrac{p_M + p_{d_i} - p_M p_{d_i}}{\sum_{i=1}^{N-C} P(E|B_i)} & i = M \\ \dfrac{p_{m_i} + p_D - p_{m_i} p_D}{\sum_{i=1}^{N-C} P(E|B_i)} & i = D \\ \dfrac{p_{m_i} + p_{d_i} - p_{m_i} p_{d_i}}{\sum_{i=1}^{N-C} P(E|B_i)} & i \neq M \text{ and } i \neq D \end{cases}$$

The entropy $H(X)$ is:

$$H(X) = \sum_{i=1}^{N-C} -p_i log_2(p_i) \tag{13}$$

so given that a collaborator of the corresponding malicious proxy node is on the path, the degree of anonymity is

$$d_C = \frac{1}{N-C} \frac{H(X|M=D)}{H_M} + \frac{N-C-1}{N-C} \frac{H(X|M \neq D)}{H_M}$$
$$= \frac{H(X|M=D) + (N-C-1)H(X|M \neq D)}{(N-C)H_M} \tag{14}$$

Furthermore, the situation where the message goes only through honest mix nodes also need to be considered. The probability is

$$P_H = \sum_{i=0}^{l} \left[(1 - p_{f\,i})(\frac{N-C}{N})^i \prod_{j=0}^{i-1} p_{f\,j}) \right] \tag{15}$$

and the degree of anonymity in such a situation will be $d_H = 1$ because the collaborators cannot obtain any information of the originator.

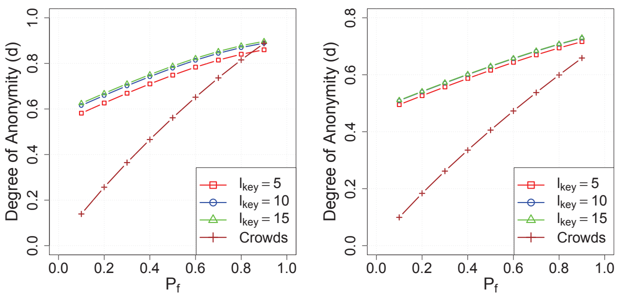
Therefore, the combined degree of anonymity

$$d = p_H + (1 - p_H) d_C \tag{16}$$

B. Comparison between different parameter settings

For illustration purposes, we compare the four primary factors influences, including N, C, l and l_{key}.

Firstly, we explore the benefits from different l_{key} and C, as shown in Fig. 2. For convenience, the horizontal axes in the figures are the forwarding probabilities of each node which are assigned consistently. In both Figs. 2a and 2b, where the percentage of collaborators is respectively 10% and 40%, the degree of anonymity is proportional to l_{key} and inversely proportional to C. But taken as a whole, our scheme can provide a much higher degree of anonymity than Crowds [5]. In addition, we can find that the effect of l_{key} gets smaller as the amount of collaborators C increases. In short, a larger l_{key} can provide users a higher degree of anonymity, but only in a proper range.

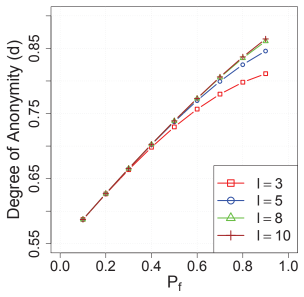


(a) $C = 50$ (b) $C = 200$

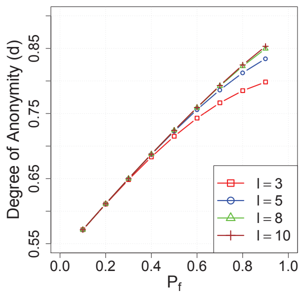
Fig. 2: Degree of Anonymity with different l_{key} and consistent P_f in each nodes ($N = 500, l = 5$)

Secondly, we compare the benefits from different l in Fig. 3a. It shows that the variation of d when $N = 500$, $C = 100$ and $l_{key} = 10$. We can discover that the degree of anonymity is in proportion to the maximum length of the forwarding path

l when $p_f > 0.5$. In spite of the fact that the effect of p_f for a particular l is unfixed, it is convergent if l is large enough. In addition, by comparing Figs. 3a and 3b, we can find that the degree of anonymity is inversely proportional to the quantity of clients when the value of C/N is fixed. But it will have no significant effect until N is more than 1000. Similar with l_{key}, the increase of l is also a promotion impact to users' privacy protection to some extent.



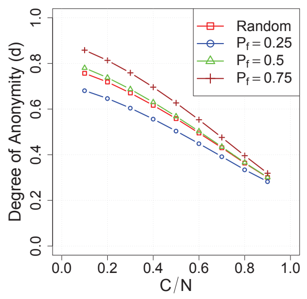
(a) $N = 500, C = 100$



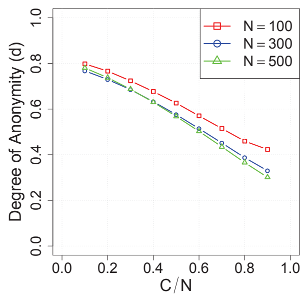
(b) $N = 1000, C = 200$

Fig. 3: Degree of Anonymity with different l and consistent P_f in each nodes ($l_{key} = 10$)

Finally, we find the relationship between the percentage of collaborators and degree of anonymity. In Fig. 4a, we make a comparison between results obtained using consistent P_f and randomly distributed P_f. Here we randomly assign l different values of forwarding probability to each node on the relaying path, and calculate an average degree of anonymity among 10000 various combinations. As shown in Fig. 4a, it is similar to the case where $P_f = 0.5$. Both Figs. 4a and 4b show that d is inversely proportional to the ratio of collaborators, but it still keeps a level higher than 0.5, in spite of the percentage of collaborators reaching up to 50%.



(a) Consistent P_f in each nodes



(b) Randomly Distributed P_f

Fig. 4: Degree of Anonymity with different $\frac{C}{N}$

V. Simulation and Evaluation

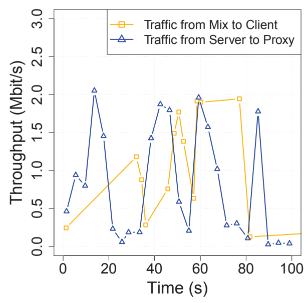
In this section, we evaluate the proposed scheme by simulative experiments using real-world user traffic data [22] to show the efficiency of our scheme, including the resistance to traffic analysis attack and the performance of web browsing.

A. Simulation Model

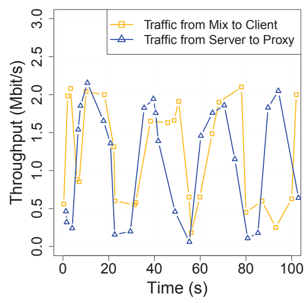
To evaluate our scheme, we construct a virtual network with 30 virtual hosts using the well-known simulator Mininet [24]. Each host is referred to as a client. The web browsing behavior log data set for Internet users is publicly available in [22], which provides 1000 randomly selected user behavior logs between May 7th and August 12th in 2012. To simulate the web browsing behavior in the data set, we run an automated HTTP browser on each client to download the index pages. Clients firstly request the HTML page and then recursively request the dependent assets, such as CSS, JS and images. All experiments run on a computer with an Intel Core 2 CPU and 2GB memory.

B. Resistance to Traffic Analysis Attack

Except the kind of adversary discussed in previous section, another adversary model that needs to be considered is the global eavesdropper. In this paper, we assume that SDN Controller is actually a powerful global eavesdropper, who can observe all the traffic flows traversing the network. Motivated by the mix-net, we employ the mixing phrase to eliminate potential correlations between the traffic entering into and leaving from an individual node. We simulate a communication between a client and a server. The client requests a page in the size of 10M from the server. During their communication, we use SDN to control the pattern of the traffic flow out of the server. Without anonymous method, the traffic pattern back into the client will be similar to the controlled pattern [12]. Then we make a comparison between effects of our mix-based design and the non-mix design, as shown in Fig. 5. The non-mix design, where nodes forward traffic without the mixing phase, exhibits high similarity between the traffic from the server and to the client node. Therefore, the adversary can infer it is the real client from its maximum similarity with the controlled pattern. The correlation between the two types of traffic flows has been disturbed by our scheme.



(a) Design in this paper

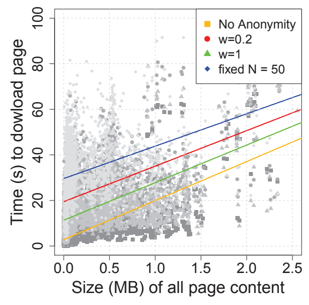


(b) Non-mix Design

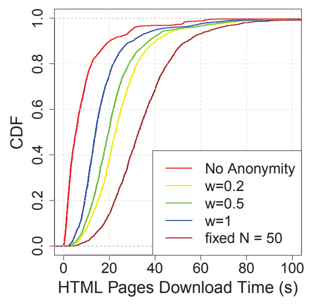
Fig. 5: Comparison between mix-based and non-mix designs

C. Web Browsing Performance

The results in Fig. 6a indicate that, it takes 20 seconds on average to download 1MB of web content with no anonymization, about 40 seconds with our scheme if the parameters of



(a) Download times



(b) CDF

Fig. 6: Download times for Web pages with different options

mix-net are fixed to $N = 50$ and $T = 3s$. Then we introduce trade-off strategy to other anonymized cases, and let the N ranges from 10 to 100, and T from 1 to 5 seconds. We make a comparison between the fixed trade-off strategies, and vary the w from 0.2 to 1. In these cases, the consume can be minimized to 25 seconds when the weight of real-time performance in the trade-off strategy is set by 1.

Fig. 6b shows a CDF of the page download times. In a non-anonymity situation, the first 50% of web pages can be downloaded within 10 seconds. While in the situation where the parameters of mix-net are fixed, the first 50% of web pages takes a little more than 30 seconds. Through the adjustment of our trade-off strategy, the download times of the first 50% of web pages can be minimized from 25 seconds to 15 seconds when w_r varies from 0.2 to 1. Furthermore, in 20 seconds, the N-fixed method only received about 30% content, while our scheme can reach up to 70%. Our scheme allows the users to flexible adjustment according to their own experience.

VI. Conclusions

In this paper, we propose a self-adaptive anonymous communication scheme to protect user's communications privacy when their network is being observed and controlled by OpenFlow Controllers. To meet the requirements on real-time performance and anonymity, we combine mix-nets and random walks method to eliminate the threat of traffic correlation analysis, and propose a self-adaptive method to maximize the real-time performance to a certain extent. Our experimental results demonstrate that our scheme can actually cover the shortages on high latency which result from mix-nets. In the future, we plan to extend our scheme to more applications and evaluate the proposed scheme in real SDN environments.

Acknowledgment

The research work reported in this paper is supported by National Science Foundation of China under Grant No. 61100172, 61272512, 61370192, 61402037, 61432015, Program for New Century Excellent Talents in University (NCET-12-0046), Beijing Natural Science Foundation No. 4132054, and Beijing Institute of Technology Research Fund Program for Young Scholars.

References

[1] Open Networking Foundation, *Software defined networking: the new norm for networks*. Technical report, 2012. https://www.opennetworking.org/images/stories/downloads/openflow/wpsdn-newnorm.pdf.

[2] N. McKeown, T. Anderson, H. Balakrishnan et al, *OpenFlow: enabling innovation in campus networks*. SIGCOMM Comput. Commun. Rev., Vol. 38(2), pp. 69-74, 2008.

[3] OpenFlow, *OpenFlow Switch Specification version 1.1.0*. Technical report 2011. http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf

[4] D. Chaum, *Untraceable electronic mail, return addresses, and digital pseudonyms*. Communications of the ACM, vol. 24, no. 2, 1981.

[5] M. Reiter and A. Rubin, *Crowds: Anonymity for Web Transactions*. ACM Transactions on Information and System Security , vol. 1, no. 1, June 1998.

[6] D. Chaum, *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability*. Journal of Cryptology, vol. 1, pp. 65-75, 1988.

[7] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, *Anonymous connections and onion routing*. IEEE Journal on Selected Areas in Communications, vol. 16, no. 4, pp. 482 - 494, 1998.

[8] R. Dingledine, N. Mathewson and P. Syverson, *Tor: The Second-Generation Onion Router*. Proceedings of the 13th USENIX Security Symposium, pp. 303-319, 2004.

[9] M. J. Freedman and R. Morris, *Tarzan: A Peer-to-Peer Anonymizing Network Layer*. Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), 2002.

[10] B. N. Levine and C. Shields, *Hordes - A Multicast Based Protocol for Anonymity*. Journal of Computer Security, vol. 10, no. 3, pp. 213-240, 2002.

[11] S. J. Murdoch and G. Danezis, *Low-Cost Traffic Analysis of Tor*. Proceedings of the 2005 IEEE Symposium on Security and Privacy, May 2005.

[12] S. Chakravarty, M. V. Barbera, G. Portokalidis, M. Polychronakis, and A. D. Keromytis, *On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records*. Proceedings of the 15th Passive and Active Measurements Conference, March 2014.

[13] P. Mittal, M. Wright, and N. Borisov, *Pisces: Anonymous Communication Using Social Networks*. Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS 2013), 2013.

[14] A. Johnson and C. Wacek, R. Jansen, M. Sherr and P. Syverson, *Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries*. Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS 2013), 2013.

[15] Tor Network Status, http://torstatus.blutmagie.de/, March 2015.

[16] S. L. Blond, D. Choffnes, W. Zhou, et al, *Towards Efficient Traffic-analysis Resistant Anonymity Networks*. SIGCOMM, pp. 303-314, August 2013.

[17] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov, *Stealthy Traffic Analysis of Low-Latency Anonymous Communication Using Throughput Fingerprinting*. Proceedings of the 18th ACM conference on Computer and Communications Security (CCS 2011), October 2011.

[18] M. AlSabah and I. Goldberg, *PCTCP: Per-Circuit TCP-over-IPsec Transport for Anonymous Communication Overlay Networks*. Proceedings of the 20th ACM conference on Computer and Communications Security (CCS 2013), 2013.

[19] C. Gülcü and G. Tsudik, *Mixing E-mail With Babel*. Proceedings of the Network and Distributed Security Symposium, NDSS '96, February 1996, pp. 2-16.

[20] G. Danezis, R. Dingledine, and N. Mathewson, *Mixminion: Design of a Type III Anonymous Remailer Protocol*. Proceedings of the 2003 IEEE Symposium on Security and Privacy, pp. 2-15, May 2003.

[21] U. Möller, L. Cottrell, P. Palfrader and L. Sassaman, *Mixmaster Protocol — Version 2*. Draft , July 2003. http://www.abditum.com/mixmaster-spec.txt

[22] CNNIC, http://cnnicdata.datatang.com, 2012.

[23] C. Diaz, S. Seys, J. Claessens and B. Preneel, *Towards measuring anonymity* Proceedings of Privacy Enhancing Technologies Workshop (PET 2002), April 2002.

[24] B. Lantz, B. Heller, and N. McKeown, *A network in a laptop: Rapid prototyping for software-defined networks*. The 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2010.