MIC: An Efficient Anonymous Communication System in Data Center Networks

Tingwei Zhu, Dan Feng, Yu Hua, Fang Wang*, Qingyu Shi, Jiahao Liu
Wuhan National Lab for Optoelectronics
School of Computer, Huazhong University of Science and Technology, Wuhan, China
*Corresponding author: wangfang@mail.hust.edu.cn

***Abstract*—With the rapid growth of application migration, the anonymity in data center networks becomes important in breaking attack chains and guaranteeing user privacy. However, existing anonymity systems are designed for the Internet environment, which suffer from high computational and network resource consumption and deliver low performance, thus failing to be directly deployed in data centers. In order to address this problem, this paper proposes an efficient and easily deployed anonymity scheme for SDN-based data centers, called MIC. The main idea behind MIC is to conceal the communication participants by modifying the source/destination addresses (such as MAC, IP and port) at switch nodes, so as to achieve anonymity. Compared with the traditional overlay-based approaches, our in-network scheme has shorter transmission paths and less intermediate operations, thus achieving higher performance with less overhead. We also propose a collision avoidance mechanism to ensure the correctness of routing, and two mechanisms to enhance the traffic-analysis resistance. Our security analysis demonstrates that MIC ensures unlinkability and improves traffic-analysis resistance. Our experiments show that MIC has extremely low overhead compared with the base-line TCP (or SSL), e.g., less than 1% overhead in terms of throughput.**

I. Introduction

With the expansion of the scale, data centers are facing a growing number of security threats from internal components (such as compromised servers, switches). According to IBM 2015 Cyber Security Intelligence Index [1], 55% of all attacks and incidents monitored by IBM in 2014 were carried out by insiders. Moreover, the outside attackers can always hack into the internal network of their targets for data breach. For example, in the data breach of Target in 2013, the attackers gain access to the Target network through stolen HVAC vendor credentials [2], and then steal 40M credit cards. As we can see, the internal network is untrustful, and more attentions should be placed on the security inside data centers.

When travelling through the untrustful network, it is important to protect the communication participants' identities and traffic patterns to conceal the activities of users. Even if the messages are encrypted, an adversary can still launch traffic-analysis attacks by examining the unencrypted information, like IP addresses, port, traffic rate or size. For example, an attacker can identify the originator and terminater of a flow by checking the source and destination addresses, and then reveal (or guess with a high probability) the ongoing operations between them by analyzing the traffic patterns. Further, the attacker can even know which user and application the communication participants belong to, as well as the scale or load of the application, through iterated traffic-analysis attacks. If the attacker aims to crash the target application or system, he can locate some key nodes of the system (like the Metadata Servers in distributed file systems) easily, and then launch active attacks, such as DoS/DDoS and worms. If he aims for data breach, this can help locate the target servers.

A lot of anonymity systems have been proposed to conceal user identity and resist traffic-analysis attacks. Such systems attempt to facilitate anonymous communication by building mix- or relay-based overlay network, such as Mixminion [3], Crowds [4], Tor [5], Dissent [6], and etc. However, these systems are designed for the Internet environment, suffering from high overhead, and cannot meet the requirements of high bandwidth and low latency in the data center environment. For example, the most popular anonymity system Tor uses layer-encrypted packets and travels through multiple indirect relays to conceal the endpoint's IP address. This approach will result in significant performance loss, since long end-to-end path length and cryptographic operations will cause high latency. Meanwhile, the indirectly traveling will incur redundant network traffic, consuming considerable network resources and reducing the total capacity of the data center network.

Most of the applications in data centers are performance sensitive, which require high bandwidth (e.g. video encoding systems) and low latency (e.g. web search systems) in transmission. Moreover, the computational and network resources are limited, and the overlay-based approaches are too expensive and will significantly reduce the capacity of the data center. Measurements show that Tor achieves 62 times higher in latency and 80% lower in throughput compared to TCP (see Figure 8 and Figure 9(a)). Therefore, it is a challenge to provide an efficient and low overhead anonymity system which is suitable for the data center environment.

The widely deployed Software Defined Networks (SDN) [7] in data centers brings new ideas for anonymous communication. The SDN architecture makes the packets forwarding more flexible, the controller can install routing rules into switches in advance and the switches modify the packet header to hide the real participants of a flow, achieving anonymous communication.

To meet to requirement of anonymity within data centers, we present Mimic Channel (MIC), an efficient in-network anonymity system designed for data center environment, with non-overlay architecture to significantly reduce resource consumption in terms of computation and network. The basic idea behind MIC is to conceal the sender and receiver of a flow by changing the addresses (such as MAC, IP and port) on multiple switches (not the host). As a result, a flow can mimic flows of other participants. A flow in MIC is called an m-flow. The switch node which changes the packet addresses is called Mimic Node (MN). The fake addresses changed by an MN are called m-addresses. An MN can be regarded as a lightweight mix or relay node in traditional anonymity systems, but is built on a switch node in the network. MIC achieves in-network anonymous communication, and hence has much shorter forwarding paths and fewer intermediate operations than traditional overlay-based schemes. Therefore, MIC is more efficient and suitable for data center environments.

However, there are two technical challenges in the MIC design. **First**, in order to achieve better anonymity, the m-addresses should be real addresses in the same network. Therefore, we need to handle the potential conflicts between two different m-flows or between an m-flow and a common flow (non-mimic flow). To avoid this routing collisions, we propose a Collision Avoidance Mechanism and design an M-Address Generation Algorithm (MAGA) to map the m-addresses of different m-flows to disjoint address spaces. **Second**, in order to increase the usability and deployability, the MIC design should not incur any modification on commodity SDN switches, as well as achieving a certain level of traffic-analysis resistance. We employ Multiple m-flows mechanism and Partial multicast mechanism to improve the traffic-analysis resistance of MIC.

The paper makes the following contributions.

- We reveal the potential security threats in non-anonymous data centers, and emphasize the importance of anonymous communication inside data centers.
- We propose an efficient anonymity scheme for SDN-based data centers, called MIC, which hides the communication participants by changing the packet header at switch nodes along transmission paths. To address the challenge of routing collision, we design a Collision Avoidance Mechanism (Sec IV-B3). We also propose two mechanisms to enhance the traffic-analysis resistance for MIC (Sec IV-C).
- We implement and evaluate MIC. Our security analysis and evaluation demonstrate that MIC can achieve session unlinkability and improve traffic-analysis resistance at low overhead, and can be easily deployed in SDN-based data centers.

The rest of this paper is organized as follow. Section II presents the background and motivation of this paper. Section III describes the system model, threat mode, goals and assumptions. Sections IV describes the MIC design. In Section V , we discuss the security of MIC. Section VI describes the implementation details and our experimental evaluation of MIC. Section VII describes the related work. Finally, we conclude our paper in Section VIII.

II. Background and Motivation

Anonymity in Data Centers. More and more enterprises deploy their business in public cloud to reduce Total Cost of Ownership (TCO) and boost service deployment. However, in many cases, the ongoing business or service, as well as their scale are a part of commercial confidentiality. This information reflects the company's decision-making and business planning. Once the information is leaked to opponents, the company will probably lose market opportunities, resulting in significant losses. Moreover, as many service deployed in cloud are online service, such as web server, they cannot tolerate any crash.

The data center faces internal security threats. An adversary can easily collect or observe a large number of traffic information at any point of the network. For example, a hacker can take over a switch by telnet attack, thereby observing and analyzing the traffic patterns to launch traffic-analysis attacks. In some server-centric network topologies, such as BCube [8], a hacker can compromise a server, and analyze the traffic passing through it. In virtualized cloud data centers, a malicious user on a guest VM can attack or compromise the host hypervisor by "guest VM escape" [9], and then can easily observe the traffic of other VMs on the same host. Much information in the packet header is useful to the adversaries, for instance, the 'ports' will typically reveal the service type (the port 80 represents Web server). In addition, there are many known shortages in existing commercial cloud. For example, Ristenpart et al. [10] points out that the internal IP addresses are statically assigned to physical machines, and one can use the internal IP address to infer the instance type and availability zone of a target service in EC2 [11]. Therefore, it is important to protect the identity of host inside the data center.

Unfortunately, traditional anonymity systems are designed for the Internet environment, which are not suitable for the data center environment. **First**, the applications in data centers have higher performance requirements than those in the Internet. All existing anonymity approaches are overlay-based, and hide the correspondence between input and output messages through hop-by-hop encryption. Therefore, they suffer from high performance overhead due to long transmission path and cryptographic operations. **Second**, the computational and networking resources are expensive in data centers. Redundant traffic in overlay architecture and multiple cryptographic operations will consume a lot of resources. Therefore, it is a challenge to achieve anonymous communication at low overhead.

Fortunately, the data center is more controllable than the Internet, and it naturally faces much less security threats than the Internet, while tolerating looser threat model. This gives us a new design space for a lightweight anonymity system.

Software-Defined Networks. The Software-Defined Network (SDN) architecture separates data plane from control plane, simplifying the network configuration, opening up the networking, and making the networking programmable. SDN

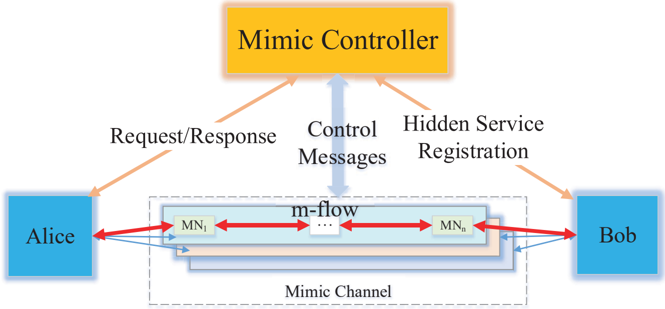


Fig. 1: The system model of MIC.

dramatically simplifies the routing in data center networks (DCNs). A lot of researches on SDN-based DCNs, such as MCTCP [12], Hedera [13] and zUpdate [14], have demonstrated the feasibility and trend of integrating SDN into the data center. In this paper, we focus on the anonymity scheme in SDN-based data centers.

III. Problem Definition

In this paper, we study the anonymity system for data centers to achieve anonymous communication and enhance traffic-analysis resistance. More specifically, an anonymity system should conceal the end-host's identity and the real traffic patterns. Taking into account the features of data centers, we study an anonymity system that can provide a practical level of anonymity at minimal performance overhead.

A. System Model

The scheme proposed in this paper is designed for SDN-based data centers, and all the switches in this paper are SDN-enabled, which can modify the packet header. MIC is a typical C/S model design, which consists of clients, MNs (Mimic Nodes) and an MC (Mimic Controller). The clients, which originate and terminate traffic, are end-hosts in data centers. The MNs are switches, which will modify the header of packets.

As shown in Figure 1, Alice is an initiator client which wants to communicate with Bob (the responder client) anonymously. She creates a transport channel between Bob and communicates with each other using MIC. A mimic channel consists of one or several end-to-end flows, called m-flows. Each m-flow travels through several MNs, which are specified by the MC. The MC, located in the SDN controller, calculates and manages the routing of each m-flow.

- The **clients**, including the initiators and the responders, can be any end-hosts in the network. An initiator establishes a mimic channel with a responder proactively before communication starts. Once a mimic channel is established, the communication pairs can exchange messages without revealing each other's identity.
- An **MN** is a lightweight mix or relay in the traditional anonymous systems, which can only modify the header of packets instead of operations like encryption/decryption, re-order, delay and batch, and etc. The commercial switches generally have no advance intelligence, and our design goal is to minimize the overhead. Any switches in the network are potential MNs.
- The **MC** is responsible for calculating and managing the routing of each m-flow. It determines the MNs in each m-flow, and generates m-addresses for each MN. With the global view of the network and each m-flow, the MC is the core of MIC design.

At a high level, MIC has two phases, the channel establishment (see in Section IV-A1) and the data forwarding (see in Section IV-A2).

B. Threat Model

The goal of adversaries is to break the unlinkability of communication pairs, seeking to infer which pairs of clients are communicating. We assume an adversary who can compromise a part of switches, the initiator client or the responder client; and who can observe some fraction of network traffic.

- **Compromising the switches:** An adversary may compromise one or a plurality of switches (but not all), which may be MNs or common switches, seeking to observe and correlate the traffic via the switches.
- **Compromising the client:** An adversary may compromise the initiator (or the responder), seeking to obtain the identity of the responder (or the initiator). For example, a hacker compromises a client in a distributed storage system, and attempts to obtain information of other nodes (like the metadata servers or storage servers), to learn which points in the network to attack next.
- **Observing the traffic:** An adversary may observe and analyze the traffic at some points in the network. For example, the switches in data centers generally have port mirroring function, which is used for Intrusion Detection System (IDS). The adversary may use the port mirroring for traffic observing, or have compromised the existing IDS.

Like most of the prior practical anonymity schemes, MIC does not protect against a global adversary who can snoop on all paths or switches. A global adversary is unlikely in practice. Specifically, it is not easy to compromise a single switch, let alone all the switches in data centers. Moreover, an IDS generally monitors the traffic from only a few ports to reduce the overhead, and the port mirroring on most switches are disabled. It is hard to observe the global traffic from all switches.

C. Goals

The main goal of MIC is to frustrate attackers from linking communication partners, achieving session unlinkability. MIC also aims to enhance resistance against size- or rate-based traffic-analysis. Moreover, MIC has the following design goals:

High performance: Most of the applications in data centers are performance sensitive, requiring high bandwidth and low latency. For example, the web services are delay sensitive

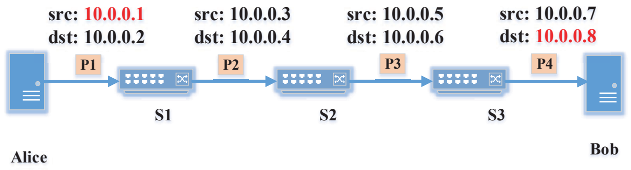


Fig. 2: An example of MIC. The intermediate switch nodes are not aware of the real 'src' 10.0.0.1 and 'dst' 10.0.0.8.

applications and the file services are bandwidth hungry applications.

Deployability: MIC design should require no kernel or switch modifications, and can deploy in common SDN-based data centers.

D. *Assumptions*

We assume the SDN controller, i.e. the MC, is secure, and all the communications between the SDN controller and the switches are secure. We believe these assumptions are reasonable. The SDN controller is the core of the network. Once the controller is compromised, the entire network will crash.

IV. Design of MIC

A. *Overview*

Similar to most of the previous anonymity systems, MIC has two phases, the channel establishment and data forwarding.

1) Channel establishment: In channel establishment, one or a set of bi-directional routing paths will be generated for each channel. Each m-flow has independent MNs and m-addresses. Specifically, when establishing a mimic channel, the initiator creates a *request* packet to the MC, and then the MC generates the corresponding routing before returning an acknowledgement to the initiator. The *request* packet contains the encrypted m-flow number, the MN number and the server address (or nickname). The MC calculates the forwarding path for each m-flow and chooses the specified number of switches as the MNs in each path.

After all paths are generated, the MC sends an acknowledgement, which contains a set of entry addresses, to the initiator. The entry address is the first m-address in an m-flow from the initiator's view, which hides the address of the responder.

2) Data Forwarding: After the mimic channel is established, the initiator or the responder can send messages anonymously through the channel. All the MNs will mimic the header of packets travel through the path to hide the participants' identities. After the communication is completed, the sender will send a notification to the MC to facilitate channel management at the MC.

We take a simple example to illustrate how an MIC works. Suppose two clients Alice (with IP address 10.0.0.1) and Bob (with IP address 10.0.0.8) are connected via three switches (S1, S2 and S3), as shown in Figure 2. For the purpose of anonymous communication, Alice dose not send messages to Bob directly, but sends a *request* to the MC for constructing an anonymous path to Alice first. After receiving the *request* from Alice, the MC calculates the forwarding path to Bob. The switches along the path modify the packet header to conceal the identity of Alice and Bob.

Specifically, suppose the packet header is denoted as a two-tuple $\langle src\_ip, dst\_ip \rangle$. The MC notifies Alice that he should send packets to the destination with address 10.0.0.2, i.e. the packet P1 is $\langle 10.0.0.1, 10.0.0.2 \rangle$. The switch S1 modifies the header of P1, and forwards it to the next hop, i.e. the packet P2 is $\langle 10.0.0.3, 10.0.0.4 \rangle$. Similarly, switches S2 and S3 modify the packet header to $\langle 10.0.0.5, 10.0.0.6 \rangle$ and $\langle 10.0.0.7, 10.0.0.8 \rangle$, respectively. It is worth noting that the last switch should modify the destination address back to the correct one, so that the receiver can handle the packets correctly without protocol stack or kernel modification.

B. *Mimic Controller*

The MC, located in the SDN controller, is the core of MIC. All the routings are calculated by the MC, and then are installed to the corresponding switches. The MC decides the forwarding path, the MNs and m-addresses for each m-flow, and has the global view of each channel. Specifically, the MC manages all the channel states, calculates and manages the routing, and handles the routing conflicts of each m-flow, ensuring the correctness of the network.

1) Channel Management: The MC needs to maintain the status of all mimic channels. When a mimic channel is constructing or the communication is finished, the initiator sends a *request* to the MC. Therefore, the MC can have the states of all m-flows.

Thus, it can be seen that the MC needs to handle a large number of establishing and shutdown *requests* in massive short communication scenes. In order to reduce the overhead on the MC, we should reuse the mimic channel among the communications between the same participants. Therefore, in these scenarios, the sender does not send shutdown *request* to the MC immediately when the communication is finished. Instead, a dedicated module in the initiator will send notification to the MC periodically.

2) Routing Calculation: MIC achieves anonymous communication by elaborate-designed routing which changes the packet header at several switches while finally leading to the right destination. The MC obtains the global view of the network and calculates all-pairs equal-cost shortest paths when initiation. After receiving the *request* packet from an initiator, the MC should generate the specified number of routing paths for m-flows.

First of all, the MC gets the initiator and the responder's addresses, the m-flow number F and the MN number N from the *request* packet. If the responder is a hidden receiver, the MC should find the address of the receiver from a hidden service map.

For each m-flow, the MC randomly selects a pre-calculated shortest path between the initiator and responder. If the path length is less than N, a new forwarding path with length larger than N will be calculated. After determining the routing path, the MC chooses N switches along the routing path as MNs. Then the MC determines the m-addresses on each MN. Finally, all the routings are installed to the corresponding switches. The MN number indicates the privacy level of a m-flow, and the more MNs will cause more overhead. We allow users to trade the privacy for performance.

3) Collision Avoidance: All the m-addresses should be in the same network name space (or subnet) to enhance the anonymity of the m-flow. Therefore, routing collision between two m-flows, or an m-flow and a common flow could happen, which will lead to errors.

Collision Example

Routing conflicts could happen when two or more flows are forwarded through the same port at a switch. The following examples show three routing conflict scenes. To simplify the description, we assume the two-tuple $< src\_ip, dst\_ip >$ identify a flow on each switch. (1) The packet addresses of two flows f_1, f_2, are changed to the same one on the same switch, as shown in Figure 3(a). (2) The packet addresses of a flow f_1 are changed to the same with another flow f_2 on the same switch, as shown in Figure 3(b). (3) The packet addresses of two flows f_1, f_2 are the same before they reach a same switch, but the switch does not change the addresses of both the two flows, as shown in Figure 3(c).

The root cause of routing conflicts is that, the m-flow will use variable addresses during communication. Therefore, an m-flow may occupy the addresses of a common flow, or two m-flows may use the same addresses simultaneously.

Collision Avoidance Mechanism

To avoid routing conflicts, we design a collision avoidance mechanism. The basic idea is to ensure each flow has a unique match entry on any switch.

First, to avoid collisions between common flows and m-flows, we use MPLS [15] label to distinguish them. Here we just use MPLS field for tagging, so that we can distinguish the flows carrying different three-tuple $\langle src\_ip, dst\_ip, mpls\rangle$. We divide the MPLS label into two disjoint categories, one used to mark the common flows (CF), and the other used to mark the m-flows (MF). Only the MC knows which MPLS labels are in CF and which are in MF. We will describe how to divide the MPLS label sets later.

Second, in order to avoid conflicts between different m-flows, we design an M-Address Generation Algorithm (MAGA). The main idea behind MAGA is to reasonably divide the address space into disjoint classes, and put the m-addresses of each m-flow (or mimic channel) into different address spaces. Therefore, for each m-flow, it can randomly select an m-address from its address space each time, avoiding collision with any other m-flows. For simplicity in description, we suppose each mimic channel contains only one m-flow. Specifically, for an m-flow, the real address is $\langle src\_ip, dst\_ip\rangle$, an MN should convert the address into m-address $\langle m\_src\_ip, m\_dst\_ip\rangle$. In order to reduce the possibility of m-address collision among different m-flows, we add MPLS label for tagging. That is, we use the three-tuples $\langle m\_src\_ip, m\_dst\_ip, mpls\rangle$ to uniquely identify an m-flow on each switch.

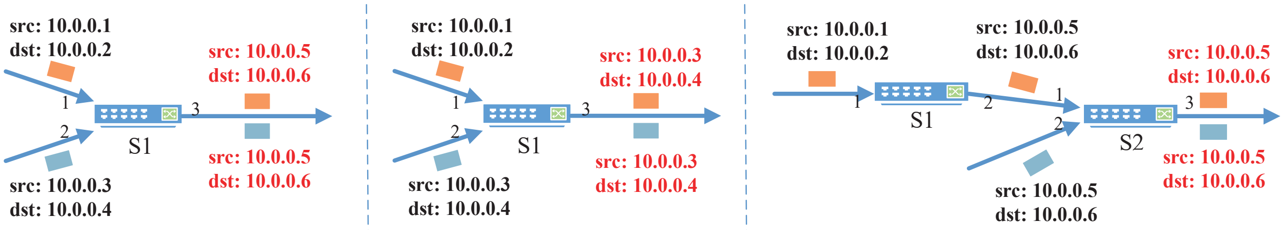
We use a hash function $f(x, y, z)$ to map the m-addresses of each m-flow to different address spaces. The hash function $f(x, y, z)$ should satisfy that, for any two different three-tuples $\langle a, b, c\rangle$ and $\langle x, y, z\rangle$, if the hash values are $f(a, b, c) = V_1$, $f(x, y, z) = V_2$, then the $V_1 \neq V_2$. In that case, given two different values V_m, V_n, we can get two disjoint three-tuple sets A_1 and A_2, which satisfy that for any $(a, b, c) \in A_1$, $(x, y, z) \in A_2$, satisfy $f(a, b, c) = V_m$ and $f(x, y, z) = V_n$, but $(a, b, c) \neq (x, y, z)$, as demonstrated in Figure 4. Therefore, if we give each m-flow a unique ID, and let any m-address (x, y, z) of an m-flow satisfies $f(x, y, z) = ID$, the routing collision between different m-flows can be avoided.

The main point is to ensure that each m-flow has a unique ID. A simple method is to monotonically increase the ID when a new m-flow arrives, and recover the expired ID when an m-flow is closed. Performed naively, a global hash function for all MNs is enough. However, in this scheme, all m-addresses (on all MNs) are constrained by a single hash function, which will result in poor security. For example, an adversary can compromise an MN, and try to find out the hash function by analyzing the m-addresses on the MN. Once an adversary knows the hash function, he can associate the packets within the same m-address space to break anonymity.

To solve the above-mentioned issues and improve anonymity of MIC, we set an independent hash function for each MN rather than a uniform hash function for all. Therefore, the adversary cannot obtain all the hash functions on all MNs easily, so making it hard to associate with the m-flows. However, as each MN has an independent hash function, we can only ensure no conflicts among m-addresses within the same MN, but not that between different MNs. Figure 3(c) shows an example of m-addresses conflict between two different MNs (if f_2 is an m-flow).

To avoid this kind of conflicts, we use the MPLS label to ensure that the m-addresses between different MNs never conflict. Again, we divide the MPLS into multiple disjoint sets, and map the MPLS sets to each MN. Therefore, the m-addresses on different MNs have different MPLS labels, which will avoid m-addresses conflicts among different MNs. To ensure anonymity, for any given MPLS label, only the MC knows which MN the label corresponds to. Similarly, we use a hash function $g(x)$ to classify the MPLS sets, and map the sets to each MN. Specifically, each MN has a unique S_ID. For an MN M whose S_ID is S, if an MPLS label m satisfies $g(m) = S$, m is in the set for M.

Thus, the key point in MAGA is to build two hash functions $f(x, y, z)$ and $g(x)$. For $f(x, y, z)$, our goal is that for a given function value V, we can get a three-tuple (a, b, c) which satisfies $f(a, b, c) = V$. Therefore, function $f(x, y, z)$ must be reversible on at least one variable. In this case, we can first determine two variables randomly, and then determine the rest variable using the inverse function, and finally get the three-tuple m-address. In order to ensure all the variable of this function are integers, we use XOR or shift operation to



(a) Two different addresses are changed to the same one.

(b) One address is changed to the same one with another.

(c) Two flows with the same address are forwarded to the same port.

Fig. 3: Examples of routing collision.

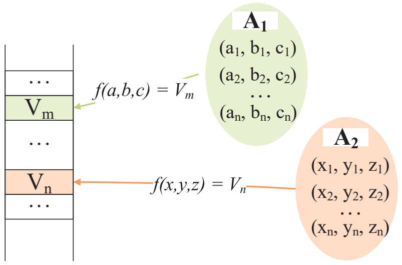


Fig. 4: Hash function demonstration.

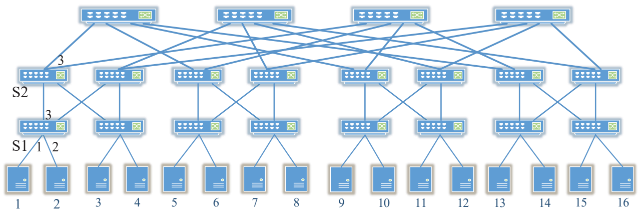


Fig. 5: Fat-tree Topology.

build the function. For example, a simple $f(x,y,z)$ can be constructed as follow.

$$\begin{aligned} f(x,y,z) =& [(x \oplus A_0) >> A_1] \oplus [(x \oplus A_2) << A_3] \\ & \oplus [(y \oplus B_0) >> B_1] \oplus [(y \oplus B_2) << B_3] \\ & \oplus [(z \oplus C_0) >> C_1] \end{aligned} \quad (1)$$

Then the inverse function for variable z is :

$$\begin{aligned} f_z^{-1}(v,x,y) =& v \oplus [(x \oplus A_0) >> A_1] \oplus [(x \oplus A_2) << A_3] \\ & \oplus [(y \oplus B_0) >> B_1] \oplus [(y \oplus B_2) << B_3] \\ & << C_1 \oplus C_0 \end{aligned} \quad (2)$$

A_0, A_1, A_2, A_2, B_0, B_1, B_2, B_3, C_0, C_1 are parameters, which can be different for different MN to build different hash functions.

To avoid an adversary distinguish the m-flows and common flows by observing the source/destination IP addresses, the m_src_ip and m_dst_ip should subject to different restrictions on different MNs. For example, for a Fat-tree topology as shown in Figure 5, the source IP of packets forward out to port 3 should be restricted to $\{1,2\}$ and $\{1,2,3,4\}$, respectively at switch $S1$ and $S2$. Meanwhile, as previously described, the MPLS label should be restricted to different sets on different MNs to avoid m-addresses conflicts among different MNs. As a result, all the three elements in $\langle m\_src\_ip, m\_dst\_ip, mpls\rangle$ cannot be arbitrarily selected. To get a three-tuple which satisfies all the restrictions quickly, we divide the MPLS to two parts $MPLS_1$ and $MPLS_2$, of which the $MPLS_1$ is subject to the restriction of distinguishing different MNs, but the $MPLS_2$ is not. Therefore, getting a satisfied three-tuple $\langle m\_src\_ip, m\_dst\_ip, mpls\rangle$ is equivalent to getting a four-tuple $\langle m\_src\_ip, m\_dst\_ip, mpls_1, mpls_2\rangle$. We construct a four variables hash function $F(\alpha,\beta,\gamma,\delta)$ and the inverse function for variable δ, $F_\delta^{-1}(v,\alpha,\beta,\gamma)$ similar to $f(x,y,z)$ and $f_z^{-1}(v,x,y)$, respectively. Finally, we first randomly select a qualifying m_src_ip, m_dst_ip, $mpls_1$, and then calculate out the $mpls_2$ using the inverse function $F_\delta^{-1}(v,\alpha,\beta,\gamma)$.

For $g(x)$, since there is only one variable, it is difficult to construct a function which meets the requirement. Hence, we divide the variable x into multiple independent variables in bits. For example, a simple solution is to divide the variable x into high bytes x_1 and low bytes x_2. Suppose the variable x has 32bits, x_1 is the high 16bits and x_2 is the low 16bits. Therefore, the function $g(x)$ is equivalent to $h(x_1,x_2)$. We can construct $h(x_1,x_2)$ and $h_{x_2}^{-1}(v,x_1)$ similar to $f(x,y,z)$ and $f_z^{-1}(v,x,y)$, respectively.

For an MN, if its S_ID is V, any MPLS label m on it should satisfy $g(m) = V$. Given the hash value V, we first randomly select the high 16bits x_1, and then calculate out the corresponding low 16bits x_2 using the inverse function $h_{x_2}^{-1}(v,x_1)$, finally the $m = x_1 << 16 + x_2$. It is worth noting that, in order to enhance security, we can make it harder for adversary to obtain the hash function by dividing the variable x in a more random way, or dividing to more sub-variables.

Similarly, for the common flows, we assign a unique function value C_ID to it, and let any m satisfy $g(m) = C\_ID$ tag the common flows.

C. Traffic-analysis Resistance

An adversary may observe and correlate the traffic at some place (switches, links or servers) in the network, seeking to find out the communication participants or what operations are processing. To enhance traffic-analysis resistance, we employ two mechanisms, multiple m-flow and partially multicast mechanism.

Multiple M-Flows Mechanism. MIC aims to achieve anonymous communication with good performance and deployability which can be deployed and used in the practical data centers. The commercial SDN switches can only process the

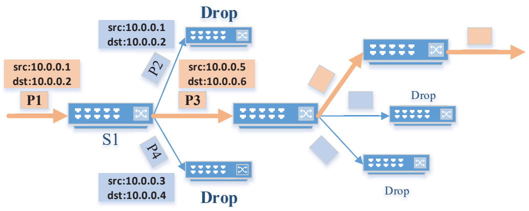


Fig. 6: Partially multicast demonstration. The MN S1 will forward out three packets P2, P3 and P4 when receiving P1, but P2 and P4 will be drop at next hop.

rules defined by southbound interfaces, like OpenFlow, but has no user-defined interfaces. Therefore, we do not delay, encrypt/decrypt or batch traffic on MNs, but just modify the packet header. To defend the size-based traffic analysis, we choose to mimic the traffic size at the source, which motivates us to employ multiple m-flows mechanism. Specifically, each mimic channel may consist of several m-flows, and each m-flow has independent routing path, MNs and m-addresses. The initiator divides the user data into slices, and each m-flow carries different amount of slices. As the traffic is divided into multiple pieces, an adversary cannot obtain the real size of the traffic unless he knows the m-flow number and has correlated all the m-flows.

Partially Multicast Mechanism. An adversary may observe all the ingress and egress traffic on an MN, and correlates the m-flow at the MN, or correlates the entire m-flow by iterated observing along the transmission path. When there are few m-flows on a specific MN, the adversary can find mismatched packet pairs in ingress and egress ports, and they have high probability to be the same packet which has been header changed by the MN. Meanwhile, since the MN processes no cryptographic operations on packets, the packets in the same m-flow look the same at each hop. An adversary can correlate with them by checking the contents of each packets. MIC cannot defeat such end-to-end correlation, but uses partially multicast mechanism to maximally decrease the success rate of this correlation. More specifically, at an MN, we will replicate the input packet to multiple packets with different m-addresses, and send the packets out from different ports simultaneously. But only one of the output packets will finally reach the receiver, the others will be dropped in the next hop, as show in Figure 6. This may be useful at the edge MNs.

D. Unlinkability

MIC achieves unlinkability by changing the packet header at multiple switches.

Sender Anonymity. MIC cannot hide the sender address if an adversary observes traffic between the sender and the first MN. However, the goal in this paper is not to provide strong anonymity at any cost, but to break the correlation between the sender and the receiver. In fact, as any switch in the network can be an MN, an adversary cannot tell whether the packets have been address modified by an MN, unless he compromises the first switch which direct links to the sender.

Receiver Anonymity. Receiver anonymity can be easily realized in MIC. Unlike the previous anonymity systems, MIC needs no additional rendezvous. The MC, which has the global view of each mimic channel can achieve the similar functionality as rendezvous or hidden service in traditional anonymity approaches. The hidden receiver first sends its contact information to the MC for anonymous service registration. The MC then adds the receiver to a hidden service map. The initiator client obtains the service name (or nickname) of the hidden receiver out of band and constructs a mimic channel using the service name. As the MC knows about the location and identify of the receiver, the channel can be constructed as normal.

V. Security Analysis

MIC is designed to achieve the communication anonymity in SDN-based data centers. We discuss a variety of attacks within our threat model and how MIC withstands them.

Compromise switches. An adversary may compromise one or several switches, which can be the common switches (non-MNs) or the MNs, along a transmission path. We consider the following situations. 1) If an adversary compromises a switch between the sender and the first MN, he can obtain the sender's address but not the receiver's; 2) If an adversary compromises a switch between the last MN and the receiver, he can obtain the receiver's address but not the sender's; 3) If an adversary compromises a switch between the first MN and the last MN, he can obtain the neither sender's nor the receiver's address. Therefore, the adversary cannot obtain both the sender and the receiver at any single point, and the global adversary is out of our threat model. As any switch in the network is likely to be an MN, an adversary cannot tell which is the first (or last) MN for a specific flow.

Compromise initiator or responder. The adversary compromises the initiator (or the responder), seeking to obtain the identity of the nodes which communicating with it, to determine the next attack target. If the responder is a hidden receiver, the initiator does not know the identity of the responder, and the responder has not idea of the initiator. Therefore, compromising the initiator (or the responder) cannot break the unlinkability of an m-flow.

Traffic observing attack. The adversary may observe (e.g. using the mirror ports of switches) the traffic on a switch, and analyze the traffic to correlate ingress and egress packets to a same flow. By iterated traffic analysis, the adversary may eventually correlate the entire m-flow. To observe the global traffic in data centers is unproductive, since the mirror port is not enabled on all switches by default. Our partially multicast mechanism helps to prevent the adversary correlating ingress and egress packets at a single MN.

Size- or rate-based traffic-analysis. The adversary may count the packet number (or size) and transmission rate at various points, seeking to analyze the traffic patterns (size or rate) of a dedicated initiator (or responder), thereby inferring what operations or businesses are processing. Our multiple m-flow mechanism can reduce the effective of this attack significantly.

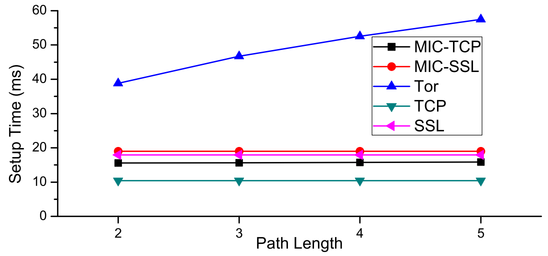


Fig. 7: Route setup time comparison among MIC, Tor, TCP and SSL.

The adversary does not know the flow number within a channel, and it is hard to correlate the flows to the same channel even if he knows the number. Even if an adversary has obtained an m-flow's traffic patterns, he cannot know the channel's traffic patterns as well.

VI. Evaluation

We build a test platform on Mininet [16]. The hardware consists of one server running Ubuntu 12.04.5 LTS operating system, with Intel (R) Xeon (R) E5-2620 @ 2.00GHz CPU, 32GB RAM. We install Mininet 2.2.0, Openvswitch 2.1.0, and Ryu 3.17 [17] on it. The network consists of 16 hosts interconnected using a Fat-tree of twenty 4-port switches, as shown in Figure 5. We evaluate the performance of MIC compared with Tor, TCP and SSL in terms of route setup latency, transmission latency and throughput. MIC-TCP and MIC-SSL in our evaluation are two MIC versions which based on TCP and SSL, respectively.

MIC implementation. MIC prototype consists of two modules: the user-end module and the MC module. We implement the user-end module on Linux platform. MIC employs typical C/S model, providing socket like programming APIs, and thus a programmer can use MIC for anonymous communication easily. We implement the MC on Ryu, a popular SDN controller platform. The communication between the client and the MC is encrypted using private key. When a client builds up a mimic channel for the first time, he should exchange a private key with the MC in advance using asymmetric encryption algorithms, like RSA [18] or D-H [19].

A. Route Setup Latency

We evaluate the route setup latency of MIC, Tor, TCP and SSL. For MIC, we measure the "MIC_connect" function time on the initiator. We use the AES function in OpenSSL for encrypting/decrypting the *request* packet. For Tor, we measure the "connect" time on the client. Specifically, we redirect the traffic to our local Tor testbed by using the "torsocks" command, and vary the route length by modifying the "DEFAULT_ROUTE_LEN" in the Tor source code. We also evaluate TCP and SSL as the base line.

Figure 7 plots the results of the route setup time varying the route length. The route length is the number of relay stages along the path in Tor, and similarly, is the number of address changing along the path in MIC. As one would expect, MIC outperforms Tor in route setup time, due to the more lightweight processing and shorter transmission path.

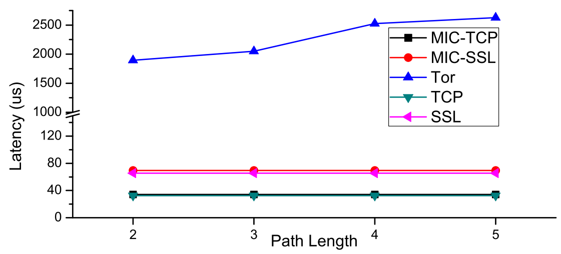


Fig. 8: Latency comparison among MIC, Tor, TCP and SSL.

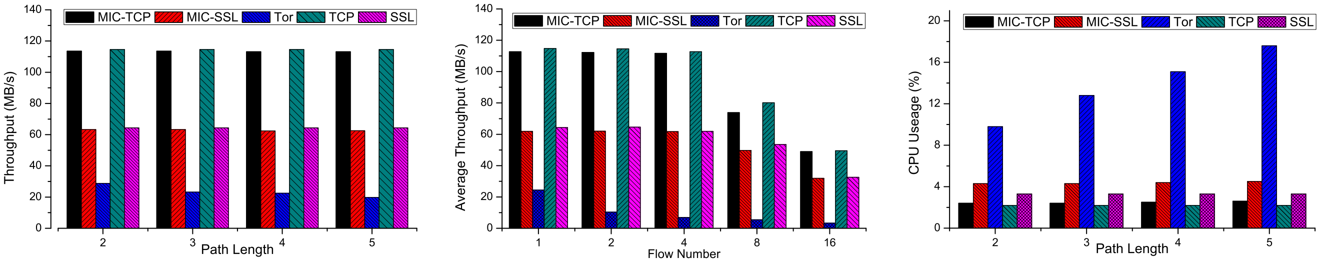
The route setup time increases with increased route length in overlay-based Tor but remains nearly the same in in-network based MIC. That is because the operations on each MN are very lightweight, and the actual length of transmission path will not increase (significantly) with increased route length. Compare to the base line TCP and SSL, MIC requires additional time for sending request to the MC, therefore, resulting a little overhead.

B. Latency and Throughput

We evaluate the latency and throughput among MIC, Tor, TCP and SSL after the session is established. In the latency evaluation, we measure the time from when the sender sends 10 bytes data to the receiver until the receiver sends 10 bytes data back. Figure 8 plots the results of latency. As can be seen from the results, MIC (including MIC-TCP and MIC-SSL) outperforms Tor significantly in terms of latency, and MIC-TCP is comparable with TCP, MIC-SSL is comparable with SSL. Compared to Tor, MIC has fewer cryptographic operations and shorter transmission path (the network paths and host protocol stacks), so that achieving lower latency. Compared to TCP (or SSL), MIC only incurs more "actions" in flow-table on MNs, whose overhead is substantially negligible.

In the throughput evaluation, we use Iperf for Tor and TCP test, and a modified Iperf for MIC and SSL. We first evaluate the throughput of one flow in different path lengths, and then evaluate the average throughput of various number of flows (the path length is set to default 3). Figure 9 (a) and (b) shows the throughput comparison among MIC, Tor, TCP and SSL. MIC achieves higher throughput than Tor due to its lightweight design. It's not a surprise to see Tor's average throughput decreases badly as the path length or flow number increases, as Tor employs the heavyweight overlay-based design. In Tor, each anonymous communication will occupy a large number of redundant network and computational resources than a common (non-anonymity) communication needs. Therefore, Tor will saturate the data center network quickly as the flow number increases, resulting in traffic congestion. However, MIC does not induce much additional length over the original (non-anonymity) path length, thereby can achieve high performance which comparable with TCP (or SSL).

We also evaluate the overall CPU usage of MIC, Tor, TCP and SSL when performing the first throughput evaluation, as shown in Figure 9 (c). The results show that the CPU overhead on MIC has a narrow increasement than TCP (or SSL) due to the extra operations on virtual switches. However, Tor suffers



(a) Throughput of one flow with different path lengths.

(b) Average throughput of various flows.

(c) CPU usage.

Fig. 9: Throughput comparison among MIC, Tor, TCP and SSL. (a) and (b) show the throughput comparison, (c) shows the CPU usage of evaluation (a).

from extremely high overhead due to the significant redundant route paths and intermediate operations.

C. Scalability Analysis

As can be seen from Sec IV-B2, the time complexity of routing calculation of MIC is $O(|F|)$, where the $|F|$ is the m-flow of a channel. By default, each MIC has one m-flow, and the m-flow number of a single MIC is generally less than 10. Thanks to our hash-based Collision Avoidance Mechanism, there is nearly no extra overhead on routing calculation compared to TCP (or SSL).

As we adopt the centralized approaches, MIC will naturally suffer from the single point failure and scalability issues. Fortunately, lots of efforts have been made on the scalability issues in SDN, such as distributed controllers [20]. MIC can be easily deployed on distributed controllers. As long as we ensure each MIC has a unique ID, our collision avoidance mechanism can guarantee the correctness of routing. Therefore, we can assign a unique ID space for each controller to make MIC work among multiple controllers.

VII. Related Work

To protect the identity of the user or service provider and defeat traffic-analysis attacks, anonymity systems has been extensively studied. Prior anonymity systems are primarily based on Mix-net [21], DC-net [22], verifiable shuffles [23] [24] or broadcast (multicast). Existing anonymity systems can be divided into two categories in accordance with the latency: high-latency anonymity systems and low-latency anonymity systems.

High-latency anonymity systems. These systems are mainly designed for applications which requires strong anonymity but can tolerate significant high latency, such as E-mail, including Babel [25], Mixmaster [26], Mixminion [3]. This systems are based on Mix-Nets, in which the messages are typically delayed for hours for batching to maximize anonymity and achieve strong traffic-analysis resistance against even a global adversary.

Low-latency anonymity systems. These systems are mainly designed for interactive applications like web browsing and Internet chat. Anonymizer [27] is the simplest low-latency anonymity system, which has only one proxy. Onion routing [28] is a real-time variant of Mix-Net in early time. Before transmitting messages, the sender picks up a list of mixs (called relays) and constructs a bi-directional circuit with the receiver via the intermediate relays. The sender layered-encrypts the messages, and each relay decrypts them then forwards them to the next hop in the circuit. Each relay knows only its previous and next hops, but has no idea of the communication participants. The second-generation Onion Routing, Tor [5] is volunteer-based, and becomes the most popular anonymity system deploy in the real word.

In P2P architecture based anonymity systems, each node can be either the traffic initiator (or recipient) or forwarder. Crowds [4] hides the traffic originator among a large number of members. MorphMix [29] makes anyone can easily join the system instead of building static mix network, and provides collision detection mechanism to identify compromised paths to enhance robust. Tarzan [30] uses cover traffic to obscure traffic patterns to defeat global observers. Aqua [31] focuses on providing high-bandwidth and strong anonymity communication for BitTorrent. Herd [32] provides scalable and traffic-analysis resistant anonymity network for VoIP systems. Hordes [33], P5 [34] and Herbivore [35] adopt multicast or broadcast mechanisms to achieve anonymity. Dissent [6] is built on DC-Net and verifiable shuffle, providing low latency, high scalability and strong anonymity. Information Slicing [36] tries to achieve anonymity communication without using public key through multi-path and secret sharing. LAP [37] provides low-latency and lightweight anonymity to protect daily online activities which are impatient to wait. Other works are focused on providing anonymity for mobile networks [38] [39], vehicular networks [40] [41], Information-Centric Networks [42] and P2P-VoD Systems [43]. As far as we are aware, MIC is the first anonymity scheme designed for data centers.

VIII. Conclusion

We present MIC, an efficient anonymity scheme aimed for data center environment. Different from the traditional overlay-based architecture, MIC adopts an in-network design, which conceals the communication participants' identifies by modifying the source/destination addresses (e.g. MAC, IP and port) at switch nodes. To address the challenge of potential routing collision in MIC, we propose a routing collision avoidance mechanism. We also propose two mechanisms, the Multiple M-flows mechanism and the Partial Multicast mechanism, to enhance the traffic-analysis resistance of MIC. As a result, we can improve anonymity of applications within data centers at negligible overhead. Experimental results show that MIC outperforms Tor significantly in performance, and is comparable with TCP (or SSL).

IX. Acknowledgments

This work is supported in part by the National High Technology Research and Development Program (863 Program) of China under Grant No.2013AA013203; National Basic Research 973 Program of China under Grant 2011CB302301; Key Laboratory of Information Storage System, Ministry of Education, China. This work is also supported by NSFC 61173043 and State Key Laboratory of Computer Architecture, No.CARCH201505.

References

[1] "Ibm security services 2015 cyber security intelligence index," http://www-03.ibm.com/security/data-breach/2015-cyber-security-index.html, 2015.

[2] "The untold story of the target attack step by step," https://aroundcyber.files.wordpress.com/2014/09/aorato-target-report.pdf, 2014.

[3] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: design of a type iii anonymous remailer protocol," in *Security and Privacy*, May 2003, pp. 2–15.

[4] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM TISSEC*, vol. 1, no. 1, pp. 66–92, nov 1998.

[5] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2004, pp. 21–21.

[6] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale," in *OSDI*. Berkeley, CA, USA: USENIX Association, 2012, pp. 179–192.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM*, vol. 38, no. 2, pp. 69–74, 2008.

[8] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," in *SIGCOMM*. New York, NY, USA: ACM, 2009, pp. 63–74.

[9] J. Kirch, "Virtual machine security guidelines version 1.0," in *The center for Internet Security*, September 2007, White Paper.

[10] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *CCS*. New York, NY, USA: ACM, 2009, pp. 199–212.

[11] "Amazon elastic compute cloud (ec2)," http://aws.amazon.com/ec2/.

[12] T. Zhu, F. Wang, Y. Hua, D. Feng, Y. Wan, Q. Shi, and Y. Xie, "MCTCP: Congestion-Aware and robust MultiCast TCP in Software-Defined networks," in *IWQoS*, Beijing, P.R. China, Jun. 2016.

[13] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *NSDI*. Berkeley, CA, USA: USENIX Association, 2010, pp. 19–19.

[14] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer, and D. Maltz, "zupdate: Updating data center networks with zero loss," in *SIGCOMM*. New York, NY, USA: ACM, 2013, pp. 411–422.

[15] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," RFC 3031 (Proposed Standard), Internet Engineering Task Force, January 2001.

[16] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *CoNEXT*. New York, NY, USA: ACM, 2012, pp. 253–264.

[17] "Ryu," http://osrg.github.io/ryu.

[18] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[19] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, no. 6, pp. 644–654, Nov. 1976. [Online]. Available: http://dx.doi.org/10.1109/TIT.1976.1055638

[20] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "Onos: Towards an open, distributed sdn os," in *HotSDN*. New York, NY, USA: ACM, 2014, pp. 1–6.

[21] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, feb 1981.

[22] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *J. Cryptol.*, vol. 1, no. 1, pp. 65–75, mar 1988.

[23] J. Furukawa and K. Sako, "An efficient scheme for proving a shuffle," in *CRYPTO*. Springer-Verlag, 2001, pp. 368–387.

[24] C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in *CCS*. New York, NY, USA: ACM, 2001, pp. 116–125.

[25] C. Gulcu and G. Tsudik, "Mixing e-mail with babel," in *Proceedings of the Symposium on Network and Distributed System Security*, Feb 1996, pp. 2–16.

[26] U. Moller and L. Cottrell, "Mixmaster protocol – version 2," Draft, January 2000., http://www.eskimo.com/ rowdenw/crypt/Mix/draft-moeller-mixmaster2-protocol-00.txt.

[27] "Anonymizer," https://www.anonymizer.com/.

[28] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding routing information," in *Proceedings of the First International Workshop on Information Hiding*. London, UK, UK: Springer-Verlag, 1996, pp. 137–150.

[29] M. Rennhard and B. Plattner, "Introducing morphmix: Peer-to-peer based anonymous internet usage with collusion detection," in *WPES*. New York, NY, USA: ACM, 2002, pp. 91–102.

[30] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *CCS*. New York, NY, USA: ACM, 2002, pp. 193–206.

[31] S. Le Blond, D. Choffnes, W. Zhou, P. Druschel, H. Ballani, and P. Francis, "Towards efficient traffic-analysis resistant anonymity networks," in *SIGCOMM*. New York, NY, USA: ACM, 2013, pp. 303–314.

[32] S. Le Blond, D. Choffnes, W. Caldwell, P. Druschel, and N. Merritt, "Herd: A scalable, traffic analysis resistant anonymity network for voip systems," in *SIGCOMM*. New York, NY, USA: ACM, 2015, pp. 639–652.

[33] B. N. Levine and C. Shields, "Hordes: A multicast based protocol for anonymity," *J. Comput. Secur.*, vol. 10, no. 3, pp. 213–240, sep 2002.

[34] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "P5 : a protocol for scalable anonymous communication," in *Proceedings on Security and Privacy Symposium*, 2002, pp. 58–70.

[35] S. Goel, M. Robson, M. Polte, and E. Sirer, "Herbivore: A scalable and efficient protocol for anonymous communication," Cornell University, Tech. Rep., 2003.

[36] S. Katti, J. Cohen, and D. Katabi, "Information slicing: Anonymity using unreliable overlays," in *NSDI*. Berkeley, CA, USA: USENIX Association, 2007, pp. 4–4.

[37] H.-C. Hsiao, T. H.-J. Kim, A. Perrig, A. Yamada, S. C. Nelson, M. Gruteser, and W. Meng, "Lap: Lightweight anonymity and privacy," in *SP*. Washington, DC, USA: IEEE Computer Society, 2012, pp. 506–520.

[38] K. Chen and H. Shen, "Fine-grained encountering information collection under neighbor anonymity in mobile opportunistic social networks," in *ICNP*, November 2015.

[39] H. Shen and L. Zhao, "Alert: An anonymous location-based efficient routing protocol in manets," *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1079–1093, June 2013.

[40] Y. Xi, K.-W. Sha, W.-S. Shi, L. Schwiebert, and T. Zhang, "Probabilistic adaptive anonymous authentication in vehicular networks," *J. Comput. Sci. Technol.*, vol. 23, no. 6, pp. 916–928, Nov. 2008.

[41] Y. Xi, W. Shi, and L. Schwiebert, "Mobile anonymity of dynamic groups in vehicular networks." *Security and Communication Networks*, vol. 1, no. 3, pp. 219–231, 2008.

[42] P. Zhang, Q. Li, and P. P. C. Lee, "Achieving content-oriented anonymity with crisp," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[43] M. Lu, P. P. C. Lee, and J. C. S. Lui, "Identity attack and anonymity protection for p2p-vod systems," in *IWQoS*, June 2011, pp. 1–9.