DAENet: Making Strong Anonymity Scale in a Fully Decentralized Network

Tianxiang Shen, Jianyu Jiang, Yunpeng Jiang, Xusheng Chen, Ji Qi, Shixiong Zhao, Fengwei Zhang, Xiapu Luo, and Heming Cui, *Member, IEEE*

Abstract—Traditional anonymous networks (e.g., Tor) are vulnerable to traffic analysis attacks that monitor the whole network traffic to determine which users are communicating. To preserve user anonymity against traffic analysis attacks, the emerging mix networks mess up the order of packets through a set of centralized and explicit shuffling nodes. However, this centralized design of mix networks is insecure against targeted DoS attacks that can completely block these shuffling nodes. In this article, we present *DAENet*, an efficient mix network that resists both targeted DoS attacks and traffic analysis attacks with a new abstraction called *Stealthy Peer-to-Peer (P2P) Network*. The *stealthy P2P network* effectively hides the shuffling nodes used in a routing path into the whole network, such that adversaries cannot distinguish specific shuffling nodes and conduct targeted DoS attacks to block these nodes. In addition, to handle traffic analysis attacks, we leverage the confidentiality and integrity protection of Intel SGX to ensure trustworthy packet shuffles at each distributed host and use multiple routing paths to prevent adversaries from tracking and revealing user identities. We show that our system is scalable with moderate latency (2.2s) when running in a cluster of 10,000 participants and is robust in the case of machine failures, making it an attractive new design for decentralized anonymous communication. DAENet 's code is released on https://github.com/hku-systems/DAENet.

Index Terms—Scalable anonymous communication, P2P network, mix network, SGX, traffic analysis attack, DoS attack

1 Introduction

The Internet allows convenient communications between users, but it also leads to great concerns about anonymity since communications can be surveilled by powerful malicious attackers such as network service providers (e.g., chatting services), Internet Service Providers and National Security Agency (NSA). These adversaries usually determine if two users are talking to each other by analyzing network communication traffics [1], [2], [3]. For example, NSA is reported to collect Internet communication (e.g., emails and voice-over-IP chats) for crime investigations [4], and such information can be misused or leaked. Worse, some governments block targeted services (e.g., Telegram) that refuse to provide user communication data [5], so that users can only use services that are under surveillance and expose their identities.

To hide user identities during network communications, more and more users turn to anonymous communication systems (e.g., Tor [6], Loopix [7]). In practice, it is desirable for an anonymous system to meet three requirements: *low-latency*, *resisting traffic analysis attacks* and *resisting targeted Denial-of-Service (DoS) attacks*. First, services that call for anonymity, such as instant messaging and online payments, usually tolerate only seconds of communication latency for interactive user experience [8], [9]. Second, powerful adversaries can conduct traffic analysis by tampering, recording, and analyzing sequences of network packets. Depending on whether the adversaries actively manipulate network states (e.g., dropping packets), traffic analysis attacks can be classified as passive attacks and active attacks. The most powerful attackers are global attackers that can monitor and manipulate network packets in the whole network [10]. Third, users in an anonymous system may be blocked by targeted Denial-of-Service (DoS) attacks from powerful attackers (e.g., governments), it is important for an anonymous system to keep serving when a portion of mission-critical components are blocked.

Traditional relay-based systems (e.g., Tor [6], AP3 [11]) are popular for anonymous communication. For instance, the Bitcoin community has long been seeking anonymity communication tools such as Tor to provision stronger client anonymity guarantees in financial transactions [12]. Specifically, these systems forward encrypted messages through several relay nodes (i.e., circuit) to hide message senders and satisfy the low-latency requirement as users can communicate through a small number of relays (e.g., three relays are usually used in Tor). However, the relay-based approach is vulnerable to global traffic analysis attacks that can manipulate and record network packets of the relay circuits [13]. Worse, relay-based anonymous systems (e.g., Tor) usually make use of centralized directory

- *Tianxiang Shen, Jianyu Jiang, Xusheng Chen, Ji Qi, Shixiong Zhao, and Heming Cui are with the Department of Computer Science, the University of Hong Kong, Hong Kong. E-mail: {txshen2, jyjiang, xschen, jqi, sxzhao, heming}@cs.hku.hk.*
- *Yunpeng Jiang is with the Department of Computer Science, South China University of Technology, Guangzhou 510006, China. E-mail: ypjiang@cs.hku.hk.*
- *Fengwei Zhang is with the Department of Computer Science, Southern University of Science and Technology, Shenzhen 518055, China. E-mail: zhangfw@sustech.edu.cn.*
- *Xiapu Luo is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: csxluo@comp.polyu.edu.hk.*

Manuscript received 16 Nov. 2019; revised 2 Sept. 2020; accepted 28 Oct. 2020. Date of publication 19 Jan. 2021; date of current version 9 July 2022. (Corresponding authors: Fengwei Zhang and Heming Cui.) Digital Object Identifier no. 10.1109/TDSC.2021.3052831

servers and are susceptible to targeted DoS attacks. The emerging shuffle-based systems (e.g., Loopix [7], Dissent [14], Karaoke [15], Riposte [16], Miranda [17]) are established to resist traffic analysis attacks. First, shuffle-based systems defend against passive traffic analysis attacks by messing up the order of user messages to hide corresponding message senders. In practice, either statistical shuffles [18] or cryptographic shuffles [19], [20] is used. To guarantee that messages are shuffled sufficiently (i.e., integrity), statistical shuffle assumes that the majority of machines for message shuffles are trustworthy [7], [21], and cryptographic shuffle requires users to verify the integrity cryptographically [16], [22]. Second, some shuffle-based systems defend against malicious packet drops by asking all users to send messages in synchronized rounds, such that any misbehaved users that drop packets will be detected quickly.

Unfortunately, existing shuffle-based systems cannot defend against targeted DoS attacks and achieve low latency at the same time. To defend against targeted DoS attacks, an anonymous system has to adopt a distributed design where each user has the same role. Without centralized servers, attackers can conduct DoS attacks against only some users, while other users can still communicate. However, it is not efficient to conduct message shuffles distributively (i.e., defending targeted DoS) with integrity. Both statistical and cryptographic shuffle usually make use of only a fixed, small number of centralized servers to conduct shuffles efficiently.

However, these fixed, centralized servers are exposed to targeted DoS attacks. Specifically, statistical shuffle makes messages go through a sequence of fixed servers (e.g., owned by mutually untrusted parties), and each server conducts shuffle separately. As these servers are fixed, it is possible to assume that the majority of them are trustworthy, and the latency is low when the number of servers used is small. However, when a statistical shuffle is applied distributively using users' machines, it has to select a group of users as shuffle nodes, and it is not possible to guarantee that the majority of the selected nodes are trustworthy. On the other hand, although the integrity of shuffles in cryptographic shuffle can be verified, the verification cost increases exponentially on the number of shuffle nodes. For example, DC-Net [23] conducts shuffles in a fully distributed manner using verifiable shuffles and all-to-all broadcasts, which incurs severe computation costs and high communication latency.

Recently, Trusted Execution Environment (TEE) such as Intel SGX has been applied in various security domains to efficiently preserve code integrity and data confidentiality [24], [25]. For example, SGX-Tor [26] is the first anonymous system that leverages SGX to hide metadata such as identifiers of routing circuits, and efficiently improves Tor's abilities for defending against various attacks (e.g., bandwidth inflation). However, SGX-Tor is still vulnerable to traffic analysis attacks and targeted DoS attacks inherited from Tor. With the integrity protection of SGX, it is possible for a shuffle-based system to shuffle messages distributively by selecting a group of trustworthy shuffle nodes, and to achieve anti-DoS and low-latency at the same time.

We present DAENet [1], the first anonymous communication system based on SGX that can meet the three desirable requirements. Specifically, all users in DAENet form a structured peer-to-peer (P2P) network with metadata (e.g., user identifier) shielded by SGX, and DAENet makes use of SGX for trustworthy message shuffles. With the help of a structured P2P network [27], [28], DAENet can achieve low-latency as messages need to go through only $log(N)$ users to reach the destination. Moreover, DAENet can defend against targeted DoS attacks that block a portion of users. This is because there are no centralized servers in DAENet, and a user can communicate through unblocked neighbors in the network. However, SGX is not the silver bullet, and DAENet still needs to handle traffic analysis attacks.

First, a structured P2P network has a static network structure, and attacks can manipulate the structure to hurt anonymity. Specifically, attackers can join as neighbors of a victim to conduct eclipse attacks. To tackle this problem, DAENet proposes a *Stealthy P2P Network* with two features. First, users in DAENet are assigned with random identities and are connected with random peers structurally. Thus, attackers cannot determine the location of a user by the user's identity and cannot manipulate the user identities to conduct eclipse attacks [29]. Second, to hide message patterns, our stealthy P2P network enforces trustworthy message shuffles that mess up the orders of input network packets at each distributed SGX-enabled host, and obliviously disseminates output packets to the neighbors of each user. With the above-mentioned designs, we prove that our stealthy P2P network produces oblivious packet transmission under passive traffic analysis attacks for all participants (Section 5.1)

Second, the static traffic patterns of a structured P2P network can leak the anonymity of users. Specifically, two users within a structured P2P network communicate through the same circuit of relays. Therefore, attackers can conduct a tagging attack [30] on the static circuit to identify the sender or receiver. We propose a *distributed dead drop* abstraction to adaptively change circuits in the network for each communication round. Specifically, two communicating participants send their messages to a randomly selected user (i.e., dead drop) using a shared secret. Then, the user exchanges the two messages' payload and sends them back. Using this approach, the attackers cannot determine one simple communicating circuit and further reveal who is communicating with whom.

We implemented DAENet with 5.2k LoC in C++ on Linux. We use Chord [27] as the implementation of our structured P2P network, as it is an efficient and popular P2P network. DAENet proposes a membership protocol that attests the SGX code integrity and assists in user *join*. Meanwhile, DAENet proposes a dialing protocol to securely initialize conversations and exchange the shared secret used for constructing a sequence of dead drops, without leaking sensitive information to adversaries. DAENet also tolerates network churn and machine failures to guarantee the liveness. We compared DAENet with Loopix [7] and Dissent [32], two state-of-art, open-sourced shuffle-based anonymous

1. DAENet is for a **D**ecentralized, **A**nonymous and **E**fficient network.

TABLE 1
Comparison of DAENet to Existing Anonymous Communication Systems

Category		Latency / Scale (#users)	Anti: Passive Traffic Analysis	Anti: Active Traffic Analysis	Anti: Targeted DoS
Relay-based	Tor [6]	0.25s $\sim$ 2.5s / 8M	$\times$	$\times$	$\times$
	SGX-Tor [26]	0.525s $\sim$ 3.15s / 819	$\times$	$\times$	$\times$
	ShadowWalker [31]	$>$ 4s / 1000	$\times$	$\times$	$\surd$
	AP3 [11]	N/A / N/A	$\times$	$\times$	$\surd$
Shuffle-based	Loopix [7]	6.8s / 500	$\surd$	$\times$	$\times$
	Riposte [16]	$>$ 3600s / N/A	$\surd$	$\surd$	$\times$
	Dissent [32]	1.3s / 500	$\surd$	$\times$	$\times$
	Atom [33]	30.0s / 1024	$\surd$	$\surd$	$\times$
	Karaoke [15]	6.0s / 16M	$\surd$	$\times$	$\times$
	DAENet	2.2s / 10,000	$\surd$	$\surd$	$\surd$

"$\surd$" indicates that the system can handle such vulnerability, while "$\times$" is on the opposite.

systems. Loopix and Dissent make use of centralized servers for layer-based shuffles and cryptographically verifiable shuffles, respectively. Our evaluations show that:

- DAENet is secure. DAENet can defend against various attacks, including passive and active traffic analysis attacks and targeted DoS attacks.
- DAENet has low latency when scaling up to a large number of users. DAENet incurs only 2.2s end-to-end latency with 10,000 participants. Compared with Loopix [7], DAENet incurs 3X $\sim$ 7X lower communication latency, yet DAENet defends against DoS attacks.

In sum, the major contribution of this paper is DAENet, the first anonymous system that meets three crucial requirements of anonymous systems: low-latency, defending against traffic analysis attacks, and defending against targeted DoS attacks. Other contributions include analysis of attacks in an SGX-based anonymous system, and extensive evaluations on DAENet's security and efficiency.

The remaining of this paper is structured as follows. Section 2 describes the background. Section 3 introduces the security goals. Section 4 describes detailed anonymous protocols. Section 5 gives a security analysis. Section 6 is the performance evaluation. Section 7 discusses limitations and future directions. Section 8 is the related work and Section 9 concludes our work.

2 Background

2.1 Anonymous Communication Systems

Existing anonymous communication systems can be classified into two categories: relay-based systems and shuffle-based systems. As shown in Table 1, we compare DAENet to prior systems from the perspective of three requirements.

Among the relay-based systems, Tor [6] is the most popular anonymous network ever deployed, with an estimated eight million daily active users [34]. Tor admits volunteer nodes to form a static routing circuit between two users, resulting in only seconds of communication latency. However, Tor is susceptible to traffic analysis attacks that monitor the whole network and de-anonymize sender identities by correlating every input and output packets [35], [36], [37]. Meanwhile, recent work also shows Tor's susceptibility to targeted DoS by conducting bandwidth amplification [38]. As an improved work of Tor, SGX-Tor [26] uses trusted computing to preserve the integrity of code and hide sensitive information of Tor components (e.g., circuit ID) in enclaves. SGX-Tor incurs slightly higher latency than Tor due to the extra overhead of entering and exiting enclaves. However, SGX-Tor inherits Tor's susceptibility to traffic analysis attacks.

Other relay-based anonymous communication systems, such as ShadowWalker [31] and AP3 [11] are built upon a structured P2P network where every node acts as both a client when sending own requests and as a proxy by forwarding requests on behalf of other nodes, eliminating the concern of targeted DoS attacks. Nevertheless, both systems cannot defend against traffic analysis attacks because the ordering of packets is still observable by traffic analyzers.

In contrast to relay-based systems, shuffle-based systems resist traffic analysis attacks, more precisely, passive traffic analysis attacks by messing up the order of input packets and output packets (i.e., shuffling). To handle active traffic analysis attacks, Riposte [16] uses Private Information Retrieval (PIR) technique to detect and stop malicious packet drops [39], [40]. However, Riposte assumes that users can tolerate its hours of latency to achieve strong anonymity, violating the low-latency requirement. Atom uses cryptographic shuffle to resist packet drops, but it also incurs high communication latency because generating and verifying Atom's zero-knowledge proofs imposes high computational and time cost [33]. Loopix [7], Dissent [32] and Karaoke [15] are three shuffle-based systems that incur reasonable communication latency. However, these systems are vulnerable to active traffic analysis attacks: by arbitrarily dropping or delaying packets in the network, adversaries can infer a specific message sender by dropping packets and observing which user receives fewer packets as expected [17]. Besides, all these systems do not provide fault-tolerance, since they use a fixed set of centralized *mix* servers to shuffle messages and require all servers to be online. Thus, these mix servers are easily targeted by DoS attacks, and all these systems will lose their liveness even only one of the mix servers is blocked by DoS attacks.

2.2 Structured Peer-to-Peer Network

A structured P2P network (e.g., Chord [41], Pastry [28]) is known for its efficient membership management, practical

fault-tolerance and fast peer lookup, making it an attractive cornerstone for building anonymous communication systems. In a structured P2P network, each participant only needs to maintain a local view of the network to extend the circuit [31]. Also, a structured P2P network has the potential to hide the roles of participants by sending dummy messages along with the links between every participant.

Specifically, a structured P2P network uses Distributed Hash Table (DHT) for peer lookup. In a DHT, nodes are assigned identifiers and a range of values they are responsible for. Nodes only have knowledge about a fraction of the network called *neighbors* which are stored in routing tables. When a node tries to lookup a value, it first checks its routing table and asks a neighbor who is numerically closest to the value. The neighbor, in turn, repeats this process. The lookup ends until the receiver that owns the value is found.

DAENet uses Chord [42], an efficient DHT scheme as the underlying communication protocol. In Chord, each participant joins the network by sending a *join* request to a known Chord node. The Chord node will assign an identifier to the participant and help the participant set up its routing table. The identifier space is pictured as a ring which wraps modulo 2^b, and b is chosen according to the scale of the network. Each participant knows only a fraction of other participants in the network. Specifically, for a participant with identifier idx, it is connected to b neighbor nodes who have the numerically closest identifier to $idx + 2^i (0 \leq i < b)$. Note that not all slots in the identifier space (i.e., $[0, 2^b)$) have to be used: each slot in the identifier space, named as S, is mapped to a participant node who has numerically closest identifier to S (i.e., $Map(S) = ClosestNode_{idx}$) by using consistent hashing [43]. In DAENet, we call the neighbor nodes of a participant *successors* and the participant itself is called the *predecessor* of all its neighbor nodes. To maintain a consistent view of membership, participants periodically send *control* messages to check the liveness of their successors and will remove inactive successors from their routing tables. Unless specifically pointed out, we denote N as the total number of participants in the network.

Although Chord facilitates efficient lookup, Chord itself does not provide anonymity guarantees because the network topology is explicit to traffic analyzers. By analyzing the entering and leaving time of network packets, traffic analyzers can link the successors and predecessors of each node and further reveal the entire topology of the network by gathering all linking information. With an explicit network topology, traffic analyzers can easily drop targeted users' packets to block its anonymous service.

2.3 Intel SGX

Intel Software Guard eXtension (SGX) [24], [25] is a popular security hardware available on commodity CPUs. It provides secure execution by putting data and code inside a container called *enclave*. The *enclave* is isolated from privileged software such as the operating system (OS), firmware and hypervisor so that the protected code and data cannot be easily tampered with or revealed from outside. The trusted (enclave) and untrusted (application) components run as isolated processes, communicating through a narrow and well-defined interface. A process running outside the enclave can invoke an SGX *ECall* to switch its execution into the enclave; a process running in an enclave can invoke an *OCall* to switch its execution outside the enclave. Besides, SGX also provides remote attestation [44] to verify that a particular piece of code is running in a genuine SGX-enabled host.

3 Overview

3.1 Threat Model

We consider sophisticated and well-resourced adversaries in the network, who attempt to determine if two participants are communicating, given that the message sender or receiver may collude with the adversaries. Therefore, we consider adversaries with two capabilities: global observation and traffic control. Confronted with such adversaries, DAENet requires at least $k \cdot \log \mathcal{N}$ honest participants to ensure complete message deliveries, where the coefficient k depends on the communication rounds of a conversation, and $\mathcal{N}$ is the total number of participants in the network. Similar to other SGX-enabled systems [26], [45], SGX firmware and the code running in SGX are trusted, SGX-related side-channel attacks (e.g., cache and timing attacks) are out of the scope of this paper.

3.2 Participants As Protocol Parties

Specifically, there are three roles in DAENet: *Relay*, *Session Node* (i.e., sender/receiver) and *Dead Drop Node*.

Relays & Session Nodes. Relays are idle participants. They do not hold any conversations with other participants and are only responsible for forwarding messages in the network, including both application messages (i.e., instant messages) and underlying P2P control messages (i.e., messages for maintaining DAENet's structural topology). In contrast to relays, session nodes are participants that hold conversations with others and keep sending application messages in multiple communication rounds. Note that a participant acts as either a relay or a session node in the network.

Dead Drop Nodes. Dead drop nodes help exchange message payload between pairs of session nodes. To initialize a set of dead drop nodes, two DAENet participants first negotiate a randomly generated shared secret through the dialing protocol (Section 4.2). The shared secret is used for generating a sequence of $DeadDrop\_keys$. Since DAENet enables deterministic *KEY-ID* mapping by building on top of Chord (Sestion 2.2), $DeadDrop\_keys$ are deterministically mapped to a series of nodes. Hence two session nodes can agree on the same sequence of dead drop nodes in the network. Note that all participants can be chosen as dead drop nodes, and the duty of a dead drop node is ephemeral and will become invalid as soon as the dead drop node completes payload exchanging in a particular communication round.

Fig. 1 shows the flow of communicating through a dead drop node in DAENet's structured P2P network. By referring to a $DeadDrop\_key$, two session nodes named Alice and Bob route their messages through several relays to a designated participant (i.e., the dead drop node). The dead drop node waits for two application messages coming and

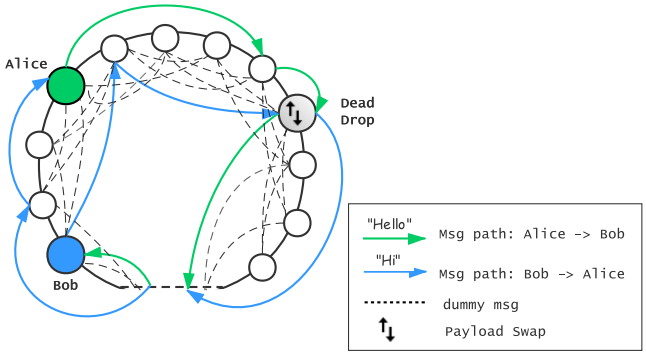


Fig. 1. An example of DAENet dead drop messaging. In a communication round, Alice and Bob separately sends two close-loop messages while exchanging their message payload at a randomly selected dead drop node.

then exchanges the message payload and sends them back to corresponding senders.

3.3 Security Goals and Defending Approaches

We demonstrate the attacks thwarted by DAENet to show the benefits of our design. Specifically, we analyze the targeted DoS attack and traffic analysis attack on DAENet and provide corresponding security analysis.

3.3.1 Defending Against Targeted DoS Attacks

Attack Assumptions. We consider an adversary who is determined to deny services to DAENet network, and we make two assumptions about the capabilities and makeup of the adversary. In particular, the adversary needs not to control a large fraction of the nodes or be able to observe the global traffic to conduct the targeted DoS attack.

First, for the capability of such attack, we assume the adversary has an attack budget $\mathcal{B}$: the adversary can deny the service of at most $\mathcal{B}$ nodes at a time. In DAENet, $\mathcal{B}$ equals $N^{1-\epsilon-\frac{1}{d}}$ - the maximum number of concurrent node failures that Chord can tolerate, in which d and ϵ are two coefficients that indicates the intensity of Chord's routing table replication scheme [46]. Second, the adversary might avoid conduct attacks from its network. Instead, the adversary can acquire (or rent) machines in public clusters to instantiate instances of DAENet participants and send dummy traffic into DAENet network, making it hard to locate the adversary.

Defending Approach. To defend against targeted DoS attacks, DAENet participants have two distinct features: equal position and ephemeral duty. First, different from prior work that uses designated authorities such as administrative servers for admitting new joining nodes, or centralized message boxes for collecting and disseminating messages from users, DAENet's participants have equal position in the network and equally act as protocol parties.

Second, the duties of roles are ephemeral. For example, DAENet uses a *dead drop* node to exchange message payload between a sender and receiver in a communication round, whereas such exchanging duty terminates as long as the communication round ends. By running participants with equal position and ephemeral duty, targeted DoS attackers cannot identify specific mission-critical nodes in the network and further block them.

3.3.2 Defending Against Passive Traffic Analysis Attacks

Attack Assumptions. Passive traffic analysis attacks intercept network packets to observe traffic patterns in order to deanonymize participants. We assume the most strong passive attacker, Global Passive Attackers (GPAs) in the network who keep eavesdropping on network traffic among all the participants and trying to find circuits of particular communications and link corresponding session nodes.

Specifically, to determine if two participants are in communication, GPAs may conduct prefix hijacking [47] to intercept network traffic and then use off-path statistical analysis [48] to sort messages. For example, in a typical passive traffic analysis attack, GPAs inspect every message of the network and keep observing the load of each participant. Since network packets' dissemination always follows the First-In-First-Out (FIFO) principle, GPAs can correlate every input and output message by recording the entering and leaving time and further restore a routing circuit. Given sufficient time, GPAs can restore all circuits for all communication sessions. Besides, GPAs can also learn the emitting rate of messages at each host. A high emitting rate might reveal a potential message sender when other parts of the network are idle.

Defending Approach. To defend against passive traffic analysis attacks from correlating any pairs of senders and receivers in conversations, our design point is to enable trustworthy message shuffling at each distributed SGX-enabled host. The shuffling process works as follows: A participant Alice maintains shuffle pools for each of its successor node. Upon receiving a message m, Alice searches for m's next-hop by conducting a Chord lookup. If the next-hop of m is the ith successor of Alice, then the message is pushed to the ith shuffle pool belonging to Alice's ith successor.

In each protocol run, Alice pulls messages from each successor's shuffle pool and sends them out with a probability p. Given a threshold α, if p is smaller than α, Alice will not pull a message from the ith successor's shuffle pool. Instead, Alice encapsulates a dummy message with the same size as a real message and sends the dummy message to its ith successor. Note that Alice may hold no messages in its ith shuffle pool at a particular protocol run. If that corner case happens, Alice needs not to pull messages from its ith shuffle pool, and will directly send a dummy message to its ith successor (Section 4.3).

3.3.3 Defending Against Active Traffic Analysis Attacks

Attack Assumptions. We assume active attackers that conduct long-term traffic analysis attacks, involving *dropping* or *delaying* packets. Such attacks have severe repercussions for anonymity guarantees of anonymous networks and are difficult to detect. For example, a *disclosure* attack in which active attackers strategically drop messages from a specific message sender allows the attacker to infer with whom the sender is communicating, by observing which participant has received fewer messages than expected [49]. We illustrate our technique to resist the *disclosure* attack because such an attack can be stealthy and hard to detect. Also, we discuss the mitigation of other aggressive active attacks that

are detectable such as traffic watermarking attacks [50] and packet hijacking attacks [51] with security analysis.

Specifically, the *disclosure* attack poses a threat to sender anonymity in DAENet: the location of a targeted sender could be revealed if active attackers collaborate with a compromised receiver and then drop messages between the targeted sender and the compromised receiver. To conduct such an attack, a compromised receiver holds a long-term connection with a targeted participant in the network and keeps sending messages to each other. During the communication, active attackers drop messages between the sender and receiver to reveal the routing circuit, based on the observation of whether the compromised receiver has received the message from the sender in time or not. We formally define such attack in Section 4.4.

Defending Approach. To defend against the *disclosure* attack, DAENet's core idea is to break the fixed circuit between two session nodes in the network by using a set of randomly generated *dead drop* nodes as the communication endpoints. In each communication round, two session nodes send their messages to a *dead drop* node and exchange corresponding messages' payload. With these *dead drop* nodes, instead of directly sending messages to each other through a fixed circuit, two session nodes send their messages to random locations in different communication rounds, thus formulating multiple different circuits in a conversation. With multiple different circuits between session nodes, the adversaries cannot reveal the location of a targeted sender by tracing back through a fixed circuit.

4 Design

This section gives a detailed discussion of DAENet's anonymous communication protocol. We start from the membership protocol that handles node join and introduce the dialing protocol to safely initialize conversations in DAENet. Then we present the design of the stealthy P2P network to defend against traffic analysis attacks.

4.1 Membership Protocol

DAENet handles node *join* by the design of the *guarder node*. When a node wants to join DAENet and uses the anonymous service, it first finds a member node through an out-of-band peer discovery service (e.g., a public forum). We call that member node the *guarder node*. A *guarder node* serves as an attestation server to verify whether an unmodified DAENet's program is executed inside a real SGX host. If the node passes the attestation, the *guarder node* replies with an automatically generated identifier, which indicates the node's location in the DAENet network.

Node Join. Specifically, node i joins DAENet with three steps. First, i creates its DAENet enclave, generates its symmetric key sk_i in the enclave and seals sk_i to local storage. Second, i sends a *join* request to the *guarder node*. The *guarder node* does a standard SGX remote attestation and succeeds with a signed report from the Intel IAS. Third, the *guarder node* verifies the report, generates an identifier of node i and encrypts it with sk_i, and sends both the sealed identifier and attestation report to node i. If node i passes the attestation, it will send a lookup request with its symmetric key sk_i to the *guarder node* to construct its routing table. The *guarder node* helps node i constructs its routing table by running a standard Chord member join protocol, and notifies a fraction of nodes that precede i that a new participant has joined the network. Note that sk_i is distributed to all the predecessors of node i, which is used for encrypting messages that are sent to node i.

Risks and Mitigation. Utilizing the above approach to admit regulated participants may have a potential risk: the channel (i.e., public forum) to join the network is public to adversaries, thus a node may discover a fake DAENet participant and join a fake DAENet which is monitored by adversaries. Also, if a malicious participant is chosen as a *guarder node* and serves as an attestation server to admit new nodes, it might refuse to admit benign nodes or try to admit specific participants (most likely be malicious).

DAENet uses mutual attestation to detect malicious *guarder nodes*. A newly joined node will also serve as an attestation server to verify the integrity of its *guarder node*. The mutual attestation is triggered when a *guarder node* sends the attestation report to node i, at the same time it provisions a self-attestation request to node i. Now node i acts as an attestation server, sends the report of the *guarder node* to Intel IAS, and waits for a signed report. Note that the attestation to a *guarder node* is hardcoded into the membership protocol and the execution is enforced unless the *guarder node* withdraws from the network. Since SGX remote attestation can help verify the integrity of the running SGX code, if a malicious *guarder node* refuses to admit benign nodes or tries to admit specific participants, the malicious *guarder node*'s code integrity is broken. Hence the malicious *guarder node* will fail to pass the attestation. The failure of passing the SGX attestation helps the new participant take actions quickly:

1) Alert users in the out-of-band peer discovery service to reduce the confidence of that malicious *guarder node*, or immediately end up contacting with that *guarder node*.
2) Retry the admission process by switching to a new *guarder node* (hopefully, one that is not malicious).

This policy limits the influence a malicious *guarder node* can do during admission, allowing DAENet to admit trustworthy participants running correct protocol. Note that DAENet can only admit SGX-enabled hosts as participants and will reject hosts without SGX.

SGX Vulnerabilities. We notice that an SGX may be compromised because of SGX vulnerabilities [52], further compromising the anonymity provided by DAENet. DAENet solves this problem by using two approaches. First, such vulnerabilities can usually be fixed through CPU microcode updates [53], and such updates increase the Security Version Number (SVN) used for attestations. DAENet's guarder node checks the latest SVN within the network and rejects nodes with SVN that is smaller than this value during attestations, such that nodes with out-of-date microcode (i.e., contain potentially compromised SGX) cannot join the network. Second, for vulnerabilities that cannot be fixed through CPU microcode updates, Intel returns a revocation certificate list during attestations. DAENet rejects attestation reports signed by these certificates and avoids the admissions of nodes with SGX vulnerabilities that cannot be fixed.

4.2 Secure Dialing: Conversation Initialization

Now that participants have joined the network, DAENet uses a secure dialing protocol to help participants initialize anonymous conversations with each other without leaking private information (e.g., identities of participants) to adversaries.

Preventing private information leakage during the initialization process is important because a service provider (namely sp) may want to keep anonymous in the network and hide its identifier from the public. If sp's identifier is public, it may become the target of DoS attacks: sp's competitors can continuously send dummy messages to sp to block its service from other benign participants.

Involving Parties. The dialing protocol involves three parties. The first party is a *client* c who wants to start a conversation with another participant in DAENet's network. The second party is a *service provider* sp who provides services (e.g., secret file sharing) to participants. Since a service provider can provide many services, a service provider typically maintains a set of *service_keys*. A service key SK_{ij} denotes the ith service provided by a service provider sp_j. The last party is a *broker* node b_j which is a designated virtual location that is responsible for receiving conversation requests to a service provider sp_j in the network.

Use Broker Node for Initialization. A typical application of DAENet is anonymous file-sharing where c tries to fetch a secret file from sp_j. Since sp_j has to hide its identifier and be reachable to others, we use a special dead drop node - the *broker* b_j to anonymously initialize conversation details without involving direct interactions between c and sp_j. The initialization mainly negotiates for three items: a shared secret sec, session ID s_{id} and an expiry time exp. sec is the seed of a pseudo-random number generator. With the same sec, c and sp agree on the same set of dead drop nodes to exchange message payload in each communication round. s_{id} is the unique identity of the conversation which is used for dead drop nodes to identify awaiting messages from the same conversation for exchanging, and exp is the longest duration for waiting for a reply (i.e., time-out).

Fig. 2 shows the complete procedure for the dialing protocol. Next, we introduce the steps from the perspective of the client and the service provider respectively.

For Client. To fetch the ith service from service provider sp_j, the client c first finds the service key SK_{ij} from an external source. The source could be a database where DAENet 's service providers put their service keys on. The client c negotiates conversation configurations with sp_j' by sending a *Register* message to sp_j's broker node b_j. The *Register* message contains the service key SK_{ij} to indicate c's requested service. The broker node b_j receives the message and verifies the contained service key to ensure a valid connection request from c. A service provider periodically asks its broker node whether there exist any *Register* messages. When sp_j finds out c's request for its ith service, it sends a configuration file to its broker node b_j. In next round, c sends a fetch request to b_j to fetch sp_j's configuration file. When c receives the configuration file, it sends an ACK to b_j to confirm a successful dialing process. By this step, the dialing process for a client is completed successfully.

For Service Provider. As a service provider, sp_j has two jobs: (1) securely assign a broker node to handle its initialization requests and (2) keep fetching initialization requests from its broker node and negotiating configuration files with clients. To complete the first job, sp_j sends an *endorsement* request to a random participant in the network to register for a broker service. If the participant replies with an acceptance, sp_j encapsulates a message which contains all the service keys it provides and sends that message to the participant. The participant then serves as the broker node b_j to handle initialization requests. To complete the second job, sp_j periodically asks b_j if there exist any initialization requests from clients. If sp_j finds any service requests, it will send the corresponding configuration file to b_j, and b_j will send the configuration file to the client. Further, sp_j tries to fetch an ACK from its broker node, if sp_j receives an ACK, then the dialing process is completed.

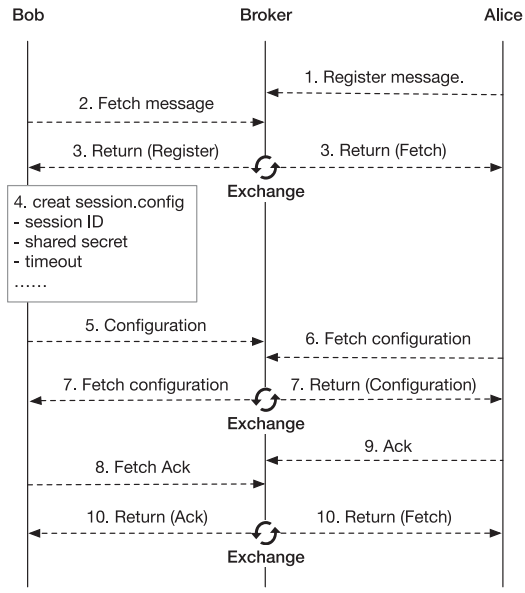


Fig. 2. Two participants of DAENet initiate their conversation through a secure dialing protocol.

Note that the existence of broker nodes for handling registration requests is not contradictory to the P2P feature of DAENet due to two reasons. First, a service provider can assign different broker nodes to serve it's registration requests, and these broker nodes are randomly distributed in the fully decentralized network. Second, the broker nodes are stealthy to the adversaries. This is because the only information the adversaries can get is the key of broker nodes. As our stealthy P2P network hides nodes' identities, the adversaries cannot locate the broker nodes in the network.

With the help of a broker node, a client registers itself to a service provider without knowing the identity of the service provider, and the service provider can securely broadcast its services and receive conversation initialization requests from the network. With a negotiated configuration file for transmission, the client and service provider can further carry out communications.

4.3 Shuffling for Sender-Receiver Unlinkability

To prevent passive traffic analysis attacks from linking two session nodes, DAENet 's shuffling strategy is designed so that, for any message that traverses a participant, adversaries cannot identify its preceding or succeeding messages and further reconstruct the entire routing circuit of a

conversation. We define *Sender-Receiver Unlinkability* as the inability for passive traffic analysis attackers to distinguish whether $\{S_{real} \rightarrow R_{real}\}$ or $\{S_{real} \rightarrow R_{other}, S_{other} \rightarrow R_{real}\}$ for a real message sender S_{real}, a real message receiver R_{real}, and other participants S_{other}, R_{other}.

Trustworthy Message Shuffling. DAENet preserves sender-receiver unlinkability with a trustworthy *shuffling* protocol. The core idea is to mess up the message orders and hide communication patterns with *dummy* messages. The shuffling protocol requires each participant to maintain *shuffle pools* for each of its successors. For each input message, Alice first decrypts the message by using its symmetric key, recalling that a sender will encrypt its messages with the symmetric key of the next hop (i.e., successor). Then Alice searches for the next hop of the message with reference to the identifier of the receiver node. If the next hop for that message is the *i*th successor of Alice, then the message is pushed to the *i*th shuffle pool belonging to Alice's *i*th successor.

In each protocol run, Alice totally pulls $\log \mathcal{N}$ messages from each of its shuffle pool by α, which is the *expected shuffle rate* - a parameter that indicates the probability of choosing a message from a shuffle pool. In other words, the *expected shuffle rate* implies whether Alice will send a message to its *i*th successor or not. If Alice does not pull a message from the shuffle pool of the *i*th successor, then Alice encapsulates a dummy message and sends the dummy message to its *i*th successor. Note that it's likely for Alice to hold no messages in its *i*th shuffle pool. In that case, Alice will directly send a dummy message to its *i*th successor.

More precisely, when Alice receives a message x, it

1) Decrypts x by using its own symmetric key key_A,
2) Discards x if x is a dummy message. Otherwise, runs the Chord lookup protocol to search for the next hop of message x. Let x_id be the identifier of the next hop, Alice resets x's message header to x_id, and pushes message x to x_id's shuffle pool.
3) Randomly pulls l messages from shuffle pools of each successor with equal probability α.
4) Encapsulates dummy message dmy_i if Alice does not pick a message from shuffle pool p_i.
5) Encrypts l messages with the symmetric key of corresponding successors and sends them out.

Denote $\mathcal{N}$ as the total number of participants in the network and k is the number of empty shuffle pools of Alice's all successors. Derived from previous statements, Alice pulls messages from $\log \mathcal{N} - k$ shuffle pools in each protocol run. The pulling process takes the form of Binomial distribution $\mathcal{X} \sim B(\log \mathcal{N}, \alpha)$ where the discrete probability α is the *expected shuffle rate*. In each round, the expected total number of real application messages and dummy messages for Alice to send is $\alpha(\log \mathcal{N} - k)$ and $k + (1 - \alpha)(\log \mathcal{N} - k)$, respectively.

Low Attack Ability. Consider the case where passive traffic analysis attackers keep observing network traffic and are capable to learn the exact number of messages in Alice's host. We define a scenario O_{x,x_1} as an adversary observing Alice's host in which message x arrives and mixes within Alice's shuffle pools. The adversary then observes $\log \mathcal{N}$ messages sending out and tries to correlate x with one of the outgoing message x_1, which is from the same conversation. Supposing the adversaries have high confidence of message x being a real message (rather than a dummy message), the following claim gives a probability on which the adversaries correctly link the previously observed message x with one of the outgoing messages x_1.

Claim 1. *Let y be the number of messages in a host in scenario O_{x,x_1}. Denote the number of non-empty shuffle pools in a node as t, and let k be the number of empty shuffle pools. After shuffling, the probability of correctly linking x to one of the outgoing message x_1 is*

$$Pr(x = x_1) = \frac{\alpha[\sum_{c=1}^{y-t-1} \frac{1}{c} Pr(C_x = c)]}{t + k}, \tag{1}$$

in which

$$Pr(C_x = c) = \binom{y-t}{c} \left(\frac{1}{t}\right)^c \left(\frac{t-1}{t}\right)^{y-t-c}. \tag{2}$$

Note that $t + k$ is the total number of outgoing messages from Alice's host. All of the outgoing messages have an equal opportunity of being the previously arrived message x, independent of the arrival time of x. This ensures that the arrival and departure time of the messages cannot be linked so that adversaries learn no sensitive information by conducting traffic analysis. Note that the probability $\frac{1}{y}$ is the upper bound for an adversary to correctly link the input message x and the corresponding output message x_1. We give an upper bound probability $\frac{1}{y}$ because all outgoing messages are from the host's shuffle pool, hence the linking probability is limited to the total number of existing messages in the current host. As there are totally y messages as we defined, the upper bound on the probability that adversaries can correctly do the traffic correlation is thus $\frac{1}{y}$. This inference applies to other shuffled-based systems that defend against traffic correlation attacks as well [7].

Thus, continuous observation of Alice's traffic leaks no sensitive information other than the present number of messages in Alice's host. We use the above claim and a security metric *likelihood* to give an end-to-end anonymity evaluation of defending passive traffic analysis attacks in security analysis (Section 5). To conclude, by randomly picking real messages from shuffle pools and disguising unpicked real messages with dummy messages, we obfuscate the adversary's view and decrease the probability of successfully correlating the input and output messages.

4.4 Hiding Sender Location From Disclosure Attacks

Attack Goal. The goal of *disclosure* attacks is to reveal the location of a targeted sender in the network. Formally, in such an attack, a malicious receiver R collaborates with active attackers who have global observations of the network to reveal the identifier of sender S. Denote a message path $||C_i|| \Leftarrow < S, P^1, P^2, ..., P^{i-1}, P^i, R >$ as the routing circuit that links the malicious receiver R and the victim sender S. Since the network topology is explicit to adversaries with a global view, R can periodically, yet slowly drops messages from its predecessors. If R drops an instant message from one of its predecessors and receives no messages

from S in next communication round, then R learns that this predecessor is P^i - the participant that acts as the previous hop of R in $||C_i||$. Now that the path $< P^i, R >$ is revealed, the adversaries try to find P^{i-1} by dropping or delaying messages from P^i's predecessors. By repeating this process, the malicious receiver R will ultimately reveal the sender S. The disclosure attack succeeds when R can receive the messages even all messages from S's predecessors are blocked.

Straw Man Approach. A straw man approach is to detect malicious disclosure behaviors in the network. However, detecting disclosure attacks in the network is difficult and inefficient. First, naively setting a threshold ϵ as *time-out* at the sender to cut off a long-term communication is impractical because we cannot determine an average latency of communication in the network as the scale of the network is unknown to each participant, and network environment differs in places. If ϵ is too large, the detection threshold is useless because attacks can still go smoothly; Otherwise, if ϵ is too small, communications become hard to carry on in the network. Second, the *loop message detection* that is used by prior work to detect malicious packet drops does *not* work in this scenario. *Loop* message is used to prove to a participant that a potentially withdrawn neighbor is online. However, since the sender does not know the exact or even relative position of the malicious receiver in the network, *loop* messages cannot tell whether the message drop is due to an offline receiver or a malicious disclosure attacker. Thus, the straw man approaches can not trivially work here.

Round-Based Dead Drop Messaging. To solve this problem, we utilize a *round-based dead drop* design to prevent malicious receivers from revealing the identifier of senders. The basic idea of this design is randomly selecting a sequence of participants in DAENet as destinations for two session nodes to exchange information in several *rounds*, and enabling full asynchrony to hide messaging patterns. Next, we discuss our *round-based dead drop* design and how we use SGX to hide the access pattern of dead drop nodes.

DAENet enforces communications through a sequence of dead drop nodes. Dead drop nodes are virtual locations where two session nodes deposit their messages (*original messages*), swap message payload from the same conversation and fetch messages (swapped messages) back. To initialize a conversation, two participants first negotiate a randomly generated *shared secret*. The *shared secret* is used for generating a sequence of *DeadDrop_keys*. The *DeadDrop_keys* are deterministically mapped to a set of nodes in the network.

Two session nodes (namely Alice and Bob) communicate with each other through these dead drop nodes. Communications happen in *rounds*. In round i, Alice and Bob independently send a message to a dead drop node N_i which is mapped from $DeadDrop\_key_i$. Each message is labeled with a *session-round* pair to indicate its unique session identity with another participant and the round of payload exchange. When N_i receives a message m, it stores it and waits for the coming of m_1 which has the same *session-round* pair as m. When m_1 arrives, N_i swaps the payload of these two messages and sends them back to corresponding message senders. The round-based dead drop messaging is effective to defend against disclosure attacks because communication circuits between session nodes change with different dead drop nodes as destinations. By splitting the static routing circuit into multiple unpredictable circuits, disclosure attackers who keep monitoring the traffic cannot reveal the previous hop in a fixed circuit by dropping messages and observing the arrival of messages.

Conversation With Compromised Nodes. Even with some fully-compromised dead drop nodes, DAENet can still preserve anonymity due to the following reasons. First, adversaries cannot determine which nodes are selected as dead drop nodes in the current conversation, and further compromise these nodes. This is because the $DeadDrop\_keys$ are generated inside SGX enclaves without involving an untrusted third-party, the locations of dead drop nodes used in the communication are kept confidential to other participants except for the session nodes, making the communication circuit unpredictable.

Second, even if the adversaries can control a fraction of nodes in the network, and these compromised nodes are happened to be selected as the dead drop nodes for a conversation, the anonymity guarantee still holds as long as one node in the circuit is honest. This is because our distributed shuffling protocol guarantees oblivious traffic pattern, and such oblivious traffic pattern offers strong anonymity against traffic analysis: a single honest participant in a circuit that correctly executes message shuffles is enough to ensure anonymity. Thus, even if all dead drop nodes are compromised, these dead drop nodes still cannot determine who is communicating with whom. Also, in section Section 5.2.1, we prove that the adversaries have low attack ability (i.e., small probability) to control all relays in a circuit when DAENet scales up.

Liveness Under Node Failures. Note that compromised dead drop nodes may not execute the payload exchange and claim to be temporarily offline. Since we cannot distinguish whether a node is failed or compromised, DAENet treats both cases as node failures. DAENet tolerates dead drop node failures with a *switch* strategy. The core idea of the strategy is that session nodes do *not* need to wait for a successfully exchanged reply from dead drop nodes in each communication round. If Alice's message m was not sent back by dead drop node N_i, Alice can resend m by switching to another *unused* dead drop node N_j with reference to $DeadDrop\_key_j$.

DAENet provides such flexibility because DAENet supports reliable datagram transfer, rather than online streaming that needs ordered messages. Thus, we assume participants can tolerate a reasonable delay of some messages and transfer other messages first when a portion of dead drop nodes fail. In the worst case when all dead drop nodes mapped from $DeadDrop\_keys$ are compromised, no successful payload exchange will take place. Since the dead drop nodes are randomly selected, the failure of all dead drop nodes indicates potential monitoring of the network. Such vulnerability will be quickly detected by the session nodes, and the session nodes are suggested to carry on their conversations later.

In addition to the *switch* strategy, to achieve privacy even with malicious dead drop nodes, DAENet leverages two policies listed as follows.

Trusted Swapping. All dead drop behaviors are executed within SGX. Since SGX guarantees the confidentiality of

decrypted messages in memory, a malicious dead drop node cannot determine whether two messages belong to the same conversation and which communication pairs are being swapped.

Ephemeral Duty. Duties in DAENet are ephemeral which means that the dead drop role does not need to persist over time. As DAENet works in asynchronous rounds, a dead drop node (agreed on by two participants) is only responsible for handling message swap in the current communication round, unless being chosen by the two participants again. Hence, a malicious dead drop node will not always hold the conversation and has no chance to reveal the link between the two participants.

5 Security Analysis

5.1 Analysis of Passive Traffic Analysis Attacks

In this subsection, we first give a theoretical proof of DAENet's oblivious messaging pattern that makes two participants in one conversation unlinkable, and then conduct a experiment to test the unlinkability under passive traffic analysis attacks with a metrics *likelihood*.

5.1.1 Theoretical Proof of Oblivious Messaging

DAENet requires a participant to send messages to all its neighbors with the same probability because a biased messaging pattern can reveal sensitive information to global passive attackers. Next, we prove that a DAENet participant sends messages to all its neighbors with the same probability and thus achieves full randomness.

Proof of Oblivious Messaging. Suppose that each node in the underlying Chord identifier ring $N <= 2^n - 1$ sends message to a random node in the ring. Each node id has $log_2 N$ neighbors, namely $id + 2^i$ for each $i <= n$. Then each neighbor of an arbitrary node id has the same expectation on access time.

Claim 3. *Each neighbor of an arbitrary node id, denoted as $id + 2^i (0 <= i < n)$, has the same number of access.*

Proof. Suppose that two node x and y are two identical nodes in the ring, we evaluate one node id, where

$$x \rightarrow \ldots \rightarrow id \rightarrow \ldots \rightarrow y. \tag{3}$$

and the identifier of node x may be equal to id.

As the routing from x to y will pass id to y, then we have $x = id - (\sum_{i=0}^{n} X_i 2^i)$, and $y = id + (\sum_{i=0}^{n} Y_i 2^i)$. X_i and Y_i is a selection variable. If a message is routing from x to y, and passes id to its i_0^{th} neighbor of node id, then $X_i = 0, i <= i_0$ and $Y_i = 0, i > i_0$. Therefore, for a neighbour $id + 2^i$ of node id, the total number of (x, y) pair that passes id and $id + 2^i$ is

$$\left(\sum_{k=0}^{n-1-i} C_{n-1-i}^{k} \right) \left(\sum_{k=0}^{i} C_{i}^{k} \right) = 2^{n-1-i} 2^i = 2^{n-1}, \tag{4}$$

which is identical to all id's neighbors. □

Limited Observable Variables. With the full randomness proved above, DAENet's protocol reveals only a small, yet insensitive set of variables to global passive attackers. First, DAENet's shuffling protocol, used for hiding communication circuits, makes all participants run in a stealthy P2P network and exposes just two variables to adversaries: the total number of sent-out messages in each round and the output rate of participants. These two variables are insensitive because they cannot reveal which participant is actually talking, as adversaries cannot distinguish an application message. Also, since we achieve full randomness of sending messages, observing the output rate does not reveal any sensitive information as well.

Second, by running code inside SGX, we prevent adversaries from directly intervening in the protocol execution and seeing the decrypted plaintext of messages. Malicious participants can monitor traffic links and deduce a set of participants' predecessors and successors under DAENet's Chord topology. However, adversaries cannot distinguish whether a received message from a predecessor is a dummy message or an application message.

5.1.2 Experimental Proof of Defending Traffic Analysis

This subsection gives an end-to-end anonymity evaluation to analyze the impact of global passive attacks in DAENet. As the strongest traffic analyzer, GPAs monitor global traffic and observe messages entering and exiting a participant, in order to link the corresponding message sender and receiver.

Thus, we analyze the unlinkability between senders and receivers by using an empirical analysis tool, used by Loopix, to study the correlation probability of two messages in the network. The security metrics that we use is called *likelihood difference*, which reveals the probability of linking a leaving message to a sender S_0 in comparison to another sender S_1. Denote the *likelihood difference* as ϵ, the two probabilities that a message is sent by S_0 and S_1 as $p_0 = Pr[S_0]$ and $p_1 = Pr[S_1]$. Our evaluated *likelihood difference* is

$$\epsilon = |log(p_0/p_1)|, \tag{5}$$

in which p_0 and p_1 can be calculated from Equations (1) and (2). To study the probabilities, we run DAENet in a local cluster, ranging from 1,024 participants to 8,192 participants that generate and send messages simultaneously with a unified messaging rate 50ms. Among the participants, 10 percent participants hold on communications while the left 90 percent participants do not communicate. We challenge the two senders S_0 and S_1 to analyze the probability: First, all participants wait for a membership warm-up time until the network becomes steady to test. All the 10 percent communication holders, except for S_0 and S_1, simultaneously send messages to the network. Then, let S_0 and S_1 encapsulate two messages, tag the two messages and send them to the network as well.

Now that there are two messages sent by S_0 and S_1 in the network which are manually labeled, while the remaining messages sent by other participants are not labeled. At each hop, we track the probability that an exiting message is labeled S_0 or S_1, and calculate the probability of being one of the senders through Theorem 2 (Section 4.3). As we pick S_0 and S_1 in their final destination, we calculate ϵ in Equation (5).

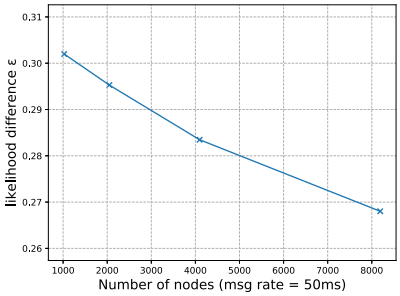


Fig. 3. Likelihood difference ϵ depending on the number of participants in the network.

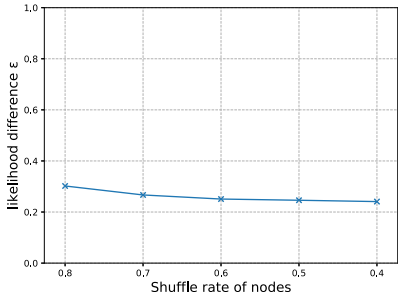


Fig. 4. Likelihood difference ϵ depending on the shuffle rate for each participant in the network.

Varying the parameter of message emitting rate and shuffle rate, we average the evaluation results over 1000 repetitions and illustrate them in Figs. 3 and 4. Our experiment shows that the expected *likelihood difference* is small (lower than 0.31).

More Participants, Stronger Anonymity. As we can see from Fig. 3, ϵ degrades almost linearly with more participants. This indicates that, by increasing the number of users of DAENet, the anonymity of participants can be further improved. When DAENet scales out to a large number of users, participants in the network process more messages. As all the messages are fully mixed in shuffle pools, the likelihood difference of two senders decreases, indicating that GPAs have less probability to link message senders and receivers.

Parameter Selection. Fig. 4 shows that the expected *likelihood difference* decreases (0.30199 to 0.2409) with decreasing shuffle rate. This figure illustrates that (1) decreasing the probability of pulling a message from shuffle pools (by decreasing the shuffle rate) with respect to the message emitting rate increases anonymity and (2) the shuffle rate has a small impact on the anonymity of participants. As the shuffler rate decreases, DAENet requires participants to send more dummy messages. To save the bandwidth cost, we consider shuffle rate = 0.8 to be a good choice in terms of anonymity.

Comparison With Loopix. Loopix also uses likelihood to evaluate its defending capability against global traffic attacks. Even if Loopix's likelihood can be smaller than DAENet, it incurs additional delay in each mix node. Specifically, in Loopix's likelihood evaluation setup (i.e., a topology of 3 layers with 3 mix nodes per layer), when Loopix achieves comparable likelihood as DAENet (0.25), it incurs an additional 1s delay in each mix node. Thus, Loopix sacrifices at least 3s latency throughout all three layers of shuffles which is larger than DAENet's end-to-end communication latency (see Section 6).

5.2 Analysis of Active Traffic Analysis Attacks

In this subsection, we analyze the impact of active traffic analysis attacks in DAENet. First, we analyze active attacks that compromise a proportion of nodes to increase the chance of choosing a fully malicious routing circuit. We continue by evaluating the security of anti-disclosure attack and other relevant active attacks.

5.2.1 Resisting Fully Controlled Circuits

Anonymous communication systems defend against active attacks with the assumption that messages will not be relayed via a fully malicious routing circuit, which is entirely controlled by the adversary. If a routing circuit is fully controlled, the adversary can trivially track all traffic and deduce that the sender and receiver are within a small anonymity set. In other words, the sender will be one of the predecessors of the entry node of the circuit, and the receiver is considered to be one of the successors of the exit node of the circuit.

Because routing circuits are chosen by the underlying P2P lookup protocol, which is enforced to execute inside SGX, the only way the adversary can succeed in conducting targeted DoS attacks is by adding more compromised nodes, in order to increase the probability of choosing compromised relays in a circuit.

Denote M_{adv} as the set of compromised nodes controlled by the adversary, N is the total number of nodes in the network and p_m as the proportion of compromised nodes. During the circuit generation process, the probability of choosing a fully malicious routing circuit is

$$Pr(circuit \in M_{adv}) \leq (p_m)^{LogN}. \quad (6)$$

Equation (6) indicates that adding more compromised nodes only slightly increases the probability of choosing a fully malicious routing circuit. When the network scales to 10,000 participants, even with a large compromised rate (p_m = 0.8 or 0.5), the probability of successfully conducting targeted DoS is less than 0.05 and 0.0001, respectively. In DAENet, even with a fully controlled routing circuit, the adversary still cannot distinguish whether a participant is talking to someone else or not. To further de-anonymize a message sender and receiver, the adversary has to make sure that a conversation indeed traverses through this fully compromised circuit, which is hard to realize in practice.

5.2.2 Resisting Aggresive Active Attacks

In this subsection, we discuss other relevant active attacks that try to de-anonymize DAENet participants.

Defeating DAENet Anti-Disclosure Protocol. As we discussed in Section 3.3, a fixed circuit in a P2P network gives chances to attackers to hierarchically reconstruct the message path. By using the dialing protocol (Section 4.2) to agree on a set of dead drop nodes in the network, DAENet prohibits adversaries from tracking an honest participant and revealing its identity. Also, since we can trust the PRNG used inside SGX to generate a series of $DeadDrop\_keys$, adversaries cannot predict every dead drop node used for exchanging message payload.

Therefore, defeating DAENet's anti-disclosure protocol requires active attackers to precisely delay all the on-path application messages in each communication round. Denote the normal averaged end-to-end communication latency as l_1, the delay time as T_d and the expected path length through dead drop as l_2. As the expected communication time through dead drop nodes is fixed, if the disclosure attacker receives a delayed message whose latency is $l_1 + (T_d \times l_2)$, then the attacker might have the confidence to reveal the message sender.

However, precisely blocking all the on-path messages is difficult and the usable attack time is short. As we will show in Section 6.1, the expectation of averaged end-to-end latency is less than $2.2s$. To successfully defeat the protocol, the adversary is supposed to precisely predict and delay all the on-path application messages with probability $1/logN$ for each link within $2.2s$, where N is the total number of participants. making it impractical to conduct.

Traffic Watermarking Attacks. Pointed out by Xinyuan [54], many proposed low-latency anonymous communication systems are vulnerable to traffic watermarking attacks. In the attack, a compromised service provider tags watermarks at messages from suspected clients, and determines if the suspected client visited the service by checking if that user has received the watermarked traffic. DAENet can defend against traffic watermarking attacks because (1) DAENet's anonymous traffic flow and the application traffic flow is mixed by trustworthy message shuffles. Thus, a watermarking attacker cannot precisely tag an application message and track that message. (2) Even if watermarking attackers can tag application messages, they cannot reveal clients because clients are not the destinations in each round of communication, instead, attackers can only reveal the set of randomly selected dead drop nodes for exchanging messages.

Aggresive Hijacking Packets. To de-anonymize network participants, a more aggressive approach is to drop a significant number of messages. For example, active attackers can launch $(n-1)$ attack [55] to track a specific message from Alice by blocking other messages to an honest participant. Also, network adversaries can inject malformed messages to replace ordinary messages. Note that in this scenario, an honest participant can easily detect such misbehavior and notice a compromised successor in the network. Honest participants can simply rejoin the network to switch to a new location and fetch a new list of neighbors for anonymous messaging.

In addition, active attackers might occasionally drop some underlying P2P control messages that are used for maintaining the membership, causing eclipse attacks that partition some nodes from the network. In that case, other nodes will lose connection with these attacked nodes and remove these attacked nodes from the routing table, which is just the same consequence as nodes are under targeted DoS attacks or failed. As a result, the partitioned nodes can simply wait for a short time and then rejoin the network.

Under Attack or Network Congestion. One possible question in DAENet is how to differentiate between message dropping due to compromised participants or network congestion. In theory, both of them can make DAENet lose its liveness while malicious message drop may also lead to privacy leakage (shown in Section 4.4). In DAENet, it is not a critical issue to differentiate between these two circumstances because DAENet is not a penalty-based system (e.g., Miranda) that makes compromised participants lose their connections in the network. On the contrary, DAENet detects messages drops to maintain a consistent view of membership in the network, caused by either misbehavior or network congestion, thus honest participants will *not* be wrongly punished.

6 Evaluation

Our evaluation was conducted on 20 computers with SGX-equipped Intel(R) Xeon(R) CPU E3-1280 v6 with 24 cores, 64 GB RAM and 2TB SSD. All computers form a cluster with 40 Gbps network. In our cluster, each machine runs multiple (up to 400) instances of DAENet client. We used Linux Traffic Control (TC) to set the network latency between clients as 40 ms to simulate the Internet environment.

We compared DAENet's performance with two state-of-art shuffle-based anonymous communication systems: Loopix and Dissent. Loopix is a popular open-sourced anonymous network that leverages Poisson-mixing shuffle strategy to protect users in the same conversation from being observed by global passive attackers, which is also guaranteed by DAENet and has been proved in Section 5.1.1. We also compared DAENet's performance with Dissent. Dissent is another open-sourced anonymous network that leverages verifiable shuffles to defend against global passive attacks. Although Dissent suffers from long-term active intersection attacks [56], it is well-known for its support of low-latency communications compared to other shuffle-based systems (e.g., Riposte, Atom). Other shuffle-based systems such as Karaoke and Vuvuzela are not evaluated because they are not open-sourced.

We built an anonymous chatting application to evaluate the performance of DAENet and our baseline systems. In our chatting application, two participants communicate with each other by sending close-loop messages through a set of dead drop nodes. To match the real-world workload of online communications, we sampled X percent of all participants as active message senders while other participants still work as normal relays. The ratio X percent is set to 10 percent by default, with reference to the Daily Active Users (DAU) of the popular WhatsApp application [57]. As Loopix has a slightly different architecture, we modified Loopix's code and wrote interfaces to forward the chatting traffic in Loopix's private cluster. Except for the client scalability evaluation, we run 50 clients on each machine (totally 1,000 clients) to evaluate the performance.

Our evaluation answers the following questions:

Section 6.1 Can DAENet support a large number of participants and provide acceptable performance?
Section 6.2 How sensitive is DAENet to its parameters?
Section 6.3 How robust is DAENet to network churn and machine failure?

6.1 Efficiency and Scalability

To analyze the efficiency and scalability of DAENet, we answer the following three questions in this subsection:

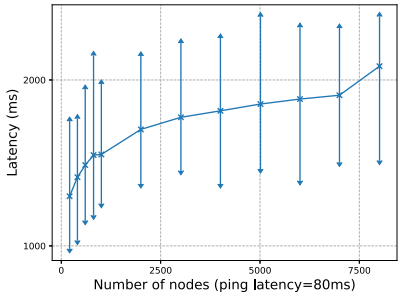


Fig. 5. Latency of DAENet when 50 to 8000 participants simultaneously send traffic at rate $\mu = 50ms$ and shuffle messages with probability $\delta = 0.8$. We assume that there is no additional delay add by participants.

- Can DAENet support a large number of users and scale horizontally?
- How does DAENet compare to prior systems?
- Will DAENet slow down the communication?

Horizontal Scalability. To demonstrate that DAENet scales horizontally, we measured the end-to-end latency for participants to route million messages as the number of participants varied. As shown in Fig. 5, the latency increases logarithmically with an increasing number of users. When 8000 participants send traffic simultaneously the latency is nearly 2000 ms.

Note that the latency overhead increases logarithmically with the total number of participants. This is because the underlying topology of DAENet is a structured P2P network, where the expected path length for one *lookup* request grows logarithmically. In DAENet we utilize Chord, the expected path length for a lookup is $log\mathcal{N}$, where $\mathcal{N}$ is the total number of participants in the network. When DAENet scales to 1M participants, the expected path length for a lookup only grows to 20.

Number of Messages. To evaluate how the number of active nodes affects latency, we increased the proportion of active nodes (i.e., nodes in communication sessions) from 10 to 95 percent, and measured the network latency, as shown in Fig. 6. As the proportion of active node increases, DAENet's latency does not increase much, while Loopix's increases dramatically. This is because Loopix incurs larger shuffle overhead with a growing number of messages through its centralized mix servers. On the contrary, participants in DAENet still send dummy messages even if there are no application messages to send, hence increasing the portion of active nodes does not produce additional network overhead because the previous idle participants just change a kind of emitted messages.

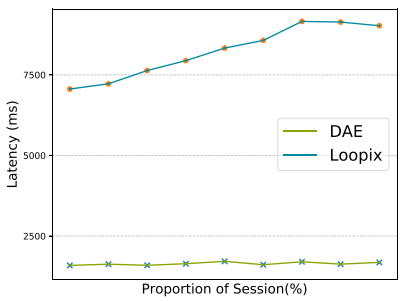


Fig. 6. Latency of DAENet for anonymous communicating for varying number of sessions. The latency does *not* increases as the number of session grows.

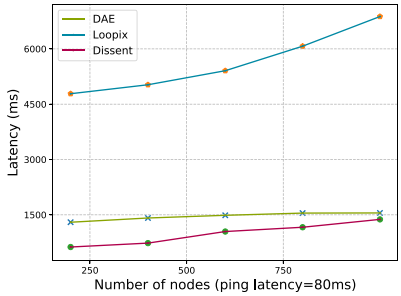


Fig. 7. Latency comparison. We measured Loopix and Dissent - two state-of-art scalable anonymous messaging systems.

Comparison to Prior Work. To compare DAENet's scalability we ran an experiment in our cluster with 20 servers. To evaluate the support for growing participants, we simulated clients by running multiple (10 $\sim$ 600) instances on each machine. For comparison, we also include the latency of Loopix and Dissent as reported in previous subsection which are the only *open-sourced* anonymous messaging system that claims to be scalable to users. We picked the system parameters $\mu = 50$ ms as the message emitting rate of participants in the network, and $\delta = 0.8$ as the shuffle rate to mix real messages and dummy messages.

Fig. 7 shows that with 800 users DAENet achieves $1.5X$ higher latency than Dissent, and $5X$ lower latency compared to Loopix. The reason why DAENet incurs higher latency than Dissent is that Dissent is a centralized system and it statically assigns servers for clients to send their messages, thus clients in Dissent doesn't need to forward messages through several hops and save time for lookups. However, such design exposes attack surface to DoS all the static servers. DAENet scales better than Loopix because all Loopix traffic must go through a single chain of servers while DAENet requires each participant to only process a fraction of messages in the network.

Latency Breakdown. To investigate DAENet's latency, we break down DAENet's latency incurred by shuffle, dead drop messaging and P2P communications, as shown in Fig. 8. Around 69.1 percent of the latency is from P2P communication, as it requires $log(N)$ steps to locate a node in the network. Dead drop communication contributes 8.4 percent of the latency. The last source of the latency, message shuffling, incurs only 22.5 percent of the latency.

As we can see from the breakdown results, DAENet will slow down the communication by adding 30.9 percent more round-trip latency. However, we believe that DAENet is useful for anonymous online communications, as participants

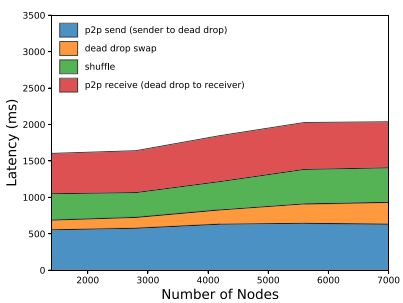


Fig. 8. Breakdown of DAENet latency.

TABLE 2
Bandwidth Cost of Running DAENet

instance/machine	20	40	60	80	100
bandwidth/instance (MB/s)	0.14	0.14	0.14	0.14	0.13

may value a stronger privacy guarantee and tolerate the moderate latency.

Bandwidth Usage. We test the bandwidth usage in a cluster of 14 machine where each machine holds several instances running independent DAENet protocol. Table 2 shows the bandwidth usage of participants running DAENet protocol. In this experiment, each test has one conversation between two randomly picked participants from all instances. To understand the minor bandwidth cost (around 0.14 MB/s), DAENet's design crucially avoids heavy usage of network resources for sending dummy messages. This is because we also add P2P control messages to the shuffle pools, such that when participants have to send out a dummy message to a neighbor, it can just replace by sending a control message rather than a dummy message. The sending of control messages in DAENet follows the rules of Chord, each participant refreshes its view of membership by sending control messages to all its neighbors every 1 second.

6.2 Parameter Sensitivity

To understand how the parameters (i.e., shuffle rate and message emitting rate) affects latency, we varied the minimum shuffle rate and message emitting rate, and measured the latency, as shown in Fig. 9 and Table 3. With a unified message emitting rate 50 ms, the latency increases dramatically when shuffle rate is decreased. This is because, in each shuffle pool of a neighbor, with a smaller shuffle rate, the probability of popping out a real message to that neighbor becomes smaller and the probability of sending a dummy message to that neighbor becomes larger. That is, a real message will have less chance to be sent out to its destination and the latency increases. Note that with a smaller shuffle rate, DAENet guarantees more obliviousness of output messages, since real messages are fully mixed with dummy messages and a malicious observer is more difficult to distinguish a real message.

When the message emitting rate increases, the latency of messages decreases because a message is popped out of the shuffle pool more quickly with a larger emitting rate. However, the descending trend of latency is smoother with a large emitting rate. This is because that the latency is also bounded by dead drop swap and P2P communication.

TABLE 3
The End-to-End Latency of DAENet With Varying Message Emitting Rate, Running in a Cluster of 1000 Nodes

Message emit rate (μs)	100	500	1000	2000	3000	5000
Latency (ms)	773	758	787	1055	1302	1594

6.3 Failure Recovery

Handling node churn is a major issue in P2P systems. To evaluate the failure resilience of DAENet, we ran DAENet for a period of time, with a typical message emitting rate 50 ms and shuffle rate 0.8, and we arbitrarily killed 10 percent of all participants three times (totally killed 30 percent active participants). The killed nodes of DAENet are sampled uniformly from existing participants, including both active participants (communicating) and idle participants. Fig. 10 shows the latency before and after killing nodes. When nodes are killed, DAENet's latency becomes extremely high because the loss of transferring message triggers timeout. After that, DAENet's latency resumes to normal in a short time, as DAENet detects the failure of messages, updates routing table and resumes processing.

7 Discussion

DAENet has two limitations. First, the current DAENet implementation does not integrate side-channel attack defenses. As SGX is susceptible to side-channel attacks where malicious software on the same platform can infer enclave data access patterns by monitoring shared resources such as caches [58], [59], it is fixable by using well-known Oblivious Ram (ORAM) algorithms, such as ZeroTrace [60].

Second, DAENet currently only supports point-to-point anonymous communication rather than anonymous broadcast, in which a participant can broadcast items to a set of receivers in an anonymous manner. This limitation forbids DAENet from supporting some security-sensitive broadcast applications such as the transaction dissemination in Bitcoin P2P network. Supporting anonymous broadcast in a P2P network could be an interesting future direction of DAENet.

8 Related Work

Tor Anonymous Network. Tor [6] is the most popular onion routing system. Due to its popularity and transparent development processes [61], many researchers have explored attacks that can de-anonymize Tor users and hidden-service providers by monitoring the network traffic. Recent attack

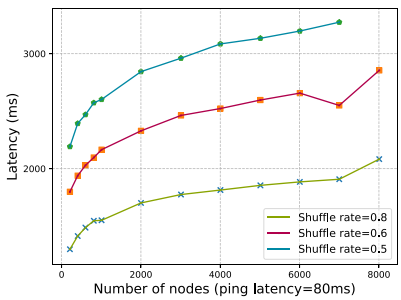


Fig. 9. The end-to-end latency of DAENet with unified message emitting rate 50ms and varying shuffle rate.

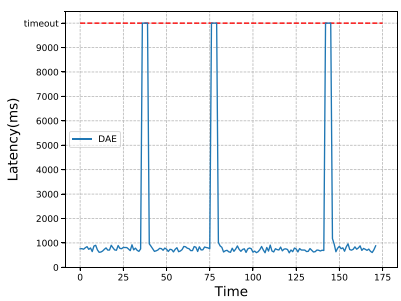


Fig. 10. Arbitrarily killing DAENet nodes to simulate network churn with a 10 percent killing rate.

vectors for Tor include BGP-based attacks [62], [63], website fingerprinting [64], [65], [66], [67], traffic correlation [36], [37], [68], [69], congestion attack [70], [71] and targeted DoS [38], [72]. Meanwhile, researchers also propose methods to enhance Tor's security by optimizing the bandwidth report for selecting guard nodes [73] and monitoring circuit construction [74]. Also, some recent Tor improvements consider generating cover traffic within middle routers of circuits, such that the middle routers can hide any relationship between compromised entry and exit nodes [75], [76].

TEE and SGX-Tor. TEE provides strong security guarantees (i.e., confidentiality and integrity) for applications with efficiency. Intel SGX [77] is one of the most popular TEE in the market. With the convenience and security properties introduced by SGX, it has been adopted for secure data analysis [45], [78], network analysis [79] and secure key-value stores [80]. SGX-Tor [81] is the first work that applies SGX to an anonymous network. As the first SGX enabled anonymous network, SGX-Tor proves the feasibility of running SGX-enabled hosts to improve an anonymous communication system's security model. As Tor relays are under the control of world-wide users, running the Tor protocol inside SGX effectively prevents malicious Tor relays from gaining private information of Tor components, such as circuit identifiers and hidden service identifiers. Although SGX-Tor mitigates many attacks against malicious Tor components, it cannot defend against network-level adversaries, potentially preventing it from being a choice of users who value strong privacy.

DAENet versus SGX-Tor. DAENet also leverages SGX to prevent private information leakage and regulate participants' behaviors. Moreover, we improve SGX-Tor's security model by protecting participants from global passive attacks and active attacks that maliciously drop and delay messages. Although DAENet incurs slightly higher end-to-end communication latency compared to SGX-Tor (shown in Table 1), we believe that users may tolerate DAENet's moderate latency to achieve stronger privacy guarantees.

Comparisons to Other Mix Networks. Vuvuzela [82] is secure against passive traffic analysis attacks. Vuvuzela's insight is to minimize the sensitive observable variables to adversaries with differential privacy techniques [83], [84]. By adding noise messages and mixing them with real messages, adversaries cannot distinguish which users are communicating. Vuvuzela requires all messages to pass through a single chain of mix servers, making it susceptible to targeted DoS attacks. In contrast, DAENet does not require a set of centralized mix servers, all messages are shuffled through each hop inside SGX.

Loopix [7] uses *cover traffic* and *Poisson mixing* mechanism to defend against passive traffic analysis attacks, and is more scalable than Vuvuzela by using parallel mix servers. Loopix observes that active attacks (e.g., (n-1) attack) can break the anonymity guarantee, and uses *loop* messages to detect such attacks. However, Loopix cannot detect a stealthy active attack that drops single messages at a time. Moreover, Loopix does not specify any after-step or how to resist other active attacks (e.g., Disclosure attack, traffic watermarking attack) whereas DAENet is secure against all these attacks.

Miranda [17] is an anonymous system that focuses on detecting active attacks in the network, including disclosure attacks and (n-1) attacks. Miranda's core idea is to build a reputation system in the network in order to measure malicious behaviors. Nevertheless, Miranda is not practical due to several simplifying assumptions: (1) a stable and synchronized network environment where operations are executed in synchronized batches, and (2) a fixed set of mix servers where a majority of them are benign. DAENet runs in an asynchronous network so that it does not need a secure clock synchronization protocol which is costly. DAENet can preserve anonymity when a majority of nodes are malicious, as long as there is one honest node in a circuit to conduct message shuffles.

Dissent [14] is based on DC-networks [85]. It protects users from being surveilled by passive traffic analysis attacks and some active attacks. Compared to DAENet, Dissent has limited scalability as it supports only several thousand nodes.

Karaoke [15] has a similar idea of using dead drop nodes to exchange messages in a mix network, and efficiently adding noise messages to hide dead drop access patterns. In the performance evaluation of Karaoke, the authors have tested Karaoke to 16 million users which is the largest evaluation scale to our best knowledge. However, Karaoke has several drawbacks that prevent it from being deployed: (1) Karaoke uses only a few mix servers to shuffle all messages in the network and requires all servers to be online, making it an attractive target of DoS attacks. DAENet shuffles messages through a group of trustworthy shuffling nodes in a fully decentralized network and provides fault-tolerance to DoS attacks. (2) Karaoke requires users to initialize conversations through out-of-band channels, which may leak sensitive information to other untrusted parties and impose unexpected bandwidth and CPU costs for clients. In contrast, DAENet handles the initialization and hides metadata during the dialing process.

Alternative Approaches. There are two approaches in the literature that have the potential to be used to enable verifiable shuffling operations in mix networks. The first approach is to use zero-knowledge proofs [86] to verify that the mix servers have correctly shuffled messages. The second approach is randomized partial checking (RPC) pointed out in the Miranda paper [17]. RPC helps detect packet drops in the network so that some active attacks can be defended with probability.

9 Conclusion

To provide practical anonymity guarantees to everyone on the Internet, anonymity networks have to develop efficient protocols to (1) accommodate a large amount of users and incur low end-to-end communication latency, and (2) provide strong anonymity guarantees against network adversaries.

As a step towards this goal, we present DAENet, the first work that enables strong anonymity in a fully decentralized network. DAENet incurs only seconds of latency when scales to 10,000 users, and is secure against targeted DoS attacks and traffic analysis attacks. We present the *stealthy P2P network* abstraction consisting two design points to efficiently preserve user anonymity. First, by using SGX to

select a group of trustworthy shuffling nodes, passive traffic analyzers cannot determine which users are communicating. Second, by safely negotiating a set of random locations (i.e., dead drops) and using these locations for exchanging message payload in each communication round, DAENet forbids disclosure attacks that track and reveal sender identifiers. We evaluated the latency and bandwidth cost of DAENet, and our evaluation results show that DAENet scales well with moderate end-to-end latency while maintaining constant-size bandwidth requirements for users.

Acknowledgments

The authors would like to thank all reviewers for their valuable comments. The work is funded by grants partly from the Huawei Innovation Research Program (HIRP) Flagship, HK RGC ECS No. 27200916, HK RGC GRF No.17207117, No. 17202318, and Croucher Innovation Award. Tianxiang Shen and Jianyu Jiang contributed equally to this work. This system was built when Jianyu Jiang was visiting the Theory Lab of Huawei Technologies, Company Ltd.

References

[1] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on tor by realistic adversaries," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 337–348.

[2] B. Resch and J. Engdegård, "Metadata time marking information for indicating a section of an audio object," Aug. 11 2015, U.S. Patent 9105300.

[3] B. Harris and R. Hunt, "TCP/IP security threats and attack methods," *Comput. Commun.*, vol. 22, no. 10, pp. 885–897, 1999.

[4] M. Sageman, "Low return on investment," *Terrorism Political Violence*, vol. 26, no. 4, pp. 614–620, 2014.

[5] P. Winter and S. Lindskog, "How the Great Firewall of China is Blocking Tor," in *Proc. 2nd USENIX Workshop Free Open Commun. Internet*, 2012, pp. 1–17.

[6] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, 2004.

[7] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The loopix anonymity system," in *Proc. 26th USENIX Secur. Symp. USENIX Secur.*, 2017, pp. 1199–1216.

[8] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measurement study of peer-to-peer file sharing systems," in *Proc. Multimedia Comput. Netw.*, 2001, pp. 156–170.

[9] M. T. Alam, H. Li, and A. Patidar, "Bitcoin for smart trading in smart grid," in *Proc. 21st IEEE Int. Workshop Local Metropolitan Area Netw.*, 2015, pp. 1/2.

[10] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proc. IEEE Symp. Secur. Privacy*, 2005, pp. 183–195.

[11] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach, "AP3: Cooperative, decentralized anonymous communication," in *Proc. 11th Workshop ACM SIGOPS Eur. Workshop*, 2004, p. 30.

[12] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2014, pp. 486–504.

[13] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of tor," in *Proc. IEEE Symp. Secur. Privacy*, 2005, pp. 183–195.

[14] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale," in *Proc. 10th USENIX Symp. Operating Syst. Des. Implementation*, 2012, pp. 179–182.

[15] D. Lazar, Y. Gilad, and N. Zeldovich, "Karaoke: Distributed private messaging immune to passive traffic analysis," in *Proc. 13th USENIX Symp. Operating Syst. Des. Implementation*, 2018, pp. 711–725.

[16] H. Corrigan-Gibbs, D. Boneh, and D. Mazières, "Riposte: An anonymous messaging system handling millions of users," in *Proc. IEEE Symp. Secur. Privacy*, 2015, pp. 321–338.

[17] H. Leibowitz, A. M. Piotrowska, G. Danezis, and A. Herzberg, "No right to remain silent: Isolating malicious mixes," in *Proc. 28th USENIX Secur. Symp. USENIX Secur.*, 2019, pp. 1841–1858.

[18] U. Parampalli, K. Ramchen, and V. Teague, "Efficiently shuffling in public," in *Proc. Int. Workshop Public Key Cryptogr.*, 2012, pp. 431–448.

[19] L. M. Sumitra and S. C. Miller, "Pathologic gambling disorder: How to help patients curb risky behavior when the future is at stake," *Postgraduate Med.*, vol. 118, no. 1, pp. 31–37, 2005.

[20] P. Paillier *et al.*, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 1999, pp. 223–238.

[21] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proc. 25th Symp. Operating Syst. Princ.*, 2015, pp. 137–152.

[22] S. Angel and S. Setty, "Unobservable communication over fully untrusted infrastructure," in *Proc. 12th USENIX Symp. Operating Syst. Des. Implementation*, 2016, pp. 551–569.

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

[24] G. Noubir and A. Sanatinia, "Trusted code execution on untrusted platforms using intel SGX," in *Proc. Virus Bull. Conf.*, 2016, pp. 1–7.

[25] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, "Malware guard extension: Using SGX to conceal cache attacks," in *Proc. Int. Conf. Detection Intrusions Malware Vulnerability Assessment*, 2017, pp. 3–24.

[26] S. Kim, J. Han, J. Ha, T. Kim, and D. Han, "SGX-Tor: A secure and practical tor anonymity network with sgx enclaves," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2174–2187, Oct. 2018.

[27] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 149–160, 2001.

[28] J. Song and S. Wang, "The pastry algorithm based on DHT," *Comput. Inf. Sci.*, vol. 2, no. 4, pp. 153–157, 2009.

[29] F. de Asís López-Fuentes, I. Eugui-De-Alba, and O. M. Ortíz-Ruiz, "Evaluating P2P networks against eclipse attacks," *Procedia Technol.*, vol. 3, pp. 61–68, 2012.

[30] K. Park and H. Lee, "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack," in *Proc. 20th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 2001, pp. 338–347.

[31] P. Mittal and N. Borisov, "ShadowWalker: Peer-to-peer anonymous communication using redundant structured topologies," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 161–172.

[32] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale," in *Proc. 10th USENIX Symp. Operating Syst. Des. Implementation*, 2012, pp. 179–182.

[33] A. Kwon, H. Corrigan-Gibbs, S. Devadas, and B. Ford, "Atom: Horizontally scaling strong anonymity," in *Proc. 26th ACM Symp. Operating Syst. Princ.*, 2017, pp. 406–422.

[34] A. Mani, T. Wilson-Brown, R. Jansen, A. Johnson, and M. Sherr, "Understanding tor usage with privacy-preserving measurement," in *Proc. Internet Meas. Conf.*, 2018, pp. 175–187.

[35] M. AlSabah and I. Goldberg, "Performance and security improvements for tor: A survey," *ACM Comput. Surv.*, vol. 49, no. 2, pp. 1–36, 2016.

[36] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on tor by realistic adversaries," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 337–348.

[37] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A new cell counter based attack against tor," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 578–589.

[38] R. Jansen, T. Vaidya, and M. Sherr, "Point break: A study of bandwidth denial-of-service attacks against tor," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 1823–1840.

[39] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proc. IEEE 36th Annu. Foundations Comput. Sci.*, 1995, pp. 41–50.

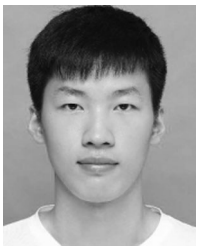
[40] R. Ostrovsky and V. Shoup, "Private information storage," in *Proc. 29th Annu. ACM Symp. Theory Comput.*, 1997, pp. 294–303.

[41] I. Stoica *et al.*, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[42] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 149–160, 2001.
[43] T. Schutt, F. Schintke, and A. Reinefeld, "Structured overlay without consistent hashing: Empirical results," in *Proc. 6th IEEE Int. Symp. Cluster Comput. Grid*, 2006, Art. no. 8.
[44] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU based attestation and sealing," in *Proc. 2nd ACM Int. Workshop Hardware Architect. Support Secur. Privacy*, 2013.
[45] F. Schuster *et al.*, "VC3: Trustworthy data analytics in the cloud using SGX," in *Proc. IEEE Symp. Secur. Privacy*, 2015, pp. 38–54.
[46] R. Kapelko, "Towards fault-tolerant chord P2P system: Analysis of some replication strategies," in *Proc. Asia-Pacific Web Conf.*, 2013, pp. 686–696.
[47] H. Ballani, P. Francis, and X. Zhang, "A study of prefix hijacking and interception in the internet," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 265–276, 2007.
[48] Y. Gilad and A. Herzberg, "Spying in the dark: TCP and Tor traffic analysis," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2012, pp. 100–119.
[49] D. Agrawal and D. Kesdogan, "Measuring anonymity: The disclosure attack," *IEEE Secur. Privacy*, vol. 1, no. 6, pp. 27–34, Nov./Dec. 2003.
[50] "Adding watermarks to images using alpha channels," [Online]. Available: http://php.net/manual/en/image.examples-watermark.php
[51] G. Danezis and L. Sassaman, "Heartbeat traffic to counter (n-1) attacks: Red-green-black mixes," in *Proc. ACM Workshop Privacy Electron. Soc.*, 2003, pp. 89–93.
[52] J. Van Bulck *et al.*, "Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 991–1008.
[53] "L1 terminal fault," [Online]. Available: https://software.intel.com/security-software-guidance/software-guidance/l1-terminal-fault
[54] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 116–130.
[55] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: Active attacks on several mix types," in *Proc. Int. Workshop Inf. Hiding*, 2002, pp. 36–52.
[56] D. Kedogan, D. Agrawal, and S. Penz, "Limits of anonymity in open environments," in *Proc. Int. Workshop Inform. Hiding*, 2002, pp. 53–69.
[57] C. Montag *et al.*, "Smartphone usage in the 21st century: Who is active on whatsapp?," *BMC Res. Notes*, vol. 8, no. 1, 2015, Art. no. 331.
[58] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi, "Software grand exposure: SGX cache attacks are practical," in *Proc. 11th USENIX Workshop Offensive Technol.*, 2017, Art. no. 11.
[59] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache attacks on intel SGX," in *Proc. 10th Eur. Workshop Syst. Secur.*, 2017, pp. 1–6.
[60] S. Sasy, S. Gorbunov, and C. W. Fletcher, "Zerotrace: Oblivious memory primitives from intel SGX," *IACR Cryptol. ePrint Arch.*, vol. 2017, 2017, Art. no. 549.
[61] T. B. Tracker, "Wiki," 2018.
[62] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal, "{RAPTOR}: Routing attacks on privacy in tor," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 271–286.
[63] Y. Sun, A. Edmundson, N. Feamster, M. Chiang, and P. Mittal, "Counter-raptor: Safeguarding tor against active routing attacks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 977–992.
[64] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 605–616.
[65] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 1187–1203.
[66] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2009, pp. 31–42.
[67] S. Li, H. Guo, and N. Hopper, "Measuring information leakage in website fingerprinting attacks and defenses," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 1977–1992.
[68] Z. Ling, J. Luo, W. Yu, X. Fu, W. Jia, and W. Zhao, "Protocol-level attacks against Tor," *Comput. Netw.*, vol. 57, no. 4, pp. 869–886, 2013.
[69] S. J. Murdoch and P. Zieliński, "Sampled traffic analysis by internet-exchange-level adversaries," in *Proc. Int. Workshop Privacy Enhancing Technol.*, 2007, pp. 167–183.
[70] N. S. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on tor using long paths," in *Proc. USENIX Secur. Symp.*, 2009, pp. 33–50.
[71] J. Geddes, R. Jansen, and N. Hopper, "How low can you go: Balancing performance with anonymity in Tor," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2013, pp. 164–184.
[72] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz, "Denial of service or denial of security?," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 92–102.
[73] R. Snader and N. Borisov, "A tune-up for Tor: Improving security and performance in the Tor network," in *Proc. NDSS*, 2008, Art. no. 127.
[74] A. Panchenko, F. Lanze, and T. Engel, "Improving performance and anonymity in the tor network," in *Proc. IEEE 31st Int. Perform. Comput. Commun. Conf.*, 2012, pp. 1–10.
[75] M. Soltani, S. Najafi, and R. Jalili, "Mid-defense: Mitigating protocol-level attacks in Tor using indistinguishability obfuscation," in *Proc. 11th Int. ISC Conf. Inf. Secur. Cryptol.*, 2014, pp. 214–219.
[76] M. Juárez, M. Imani, M. Perry, C. Dıaz, and M. Wright, "WTF-PAD: Toward an efficient website fingerprinting defense for Tor," 2015, *CoRR, abs/1512.00524*.
[77] Intel, "Software guard extensions programming reference,".
[78] W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Opaque: An oblivious and encrypted distributed analytics platform," in *Proc. 14th USENIX Conf. Netw. Syst. Des. Implementation*, 2017, pp. 283–298.
[79] M.-W. Shih, M. Kumar, T. Kim, and A. Gavrilovska, "S-NFV: Securing NFV states by using SGX," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, 2016, pp. 45–48.
[80] C. Priebe, K. Vaswani, and M. Costa, "EnclaveDB: A secure database using SGX," in *Proc. IEEE Symp. Secur. Privacy*, 2018.
[81] S. M. Kim, J. Han, J. Ha, T. Kim, and D. Han, "Enhancing security and privacy of Tor's ecosystem by using trusted execution environments," in *Proc. NSDI*, 2017, pp. 145–161.
[82] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proc. 25th ACM Symp. Operating Syst. Princ.*, 2015, pp. 137–152.
[83] C. Clifton and T. Tassa, "On syntactic anonymity and differential privacy," in *Proc. IEEE 29th Int. Conf. Data Eng. Workshops*, 2013, pp. 88–93.
[84] T. O. Li *et al.*, "Upa: An automated, accurate and efficient differentially private big-data mining system," in *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2020, pp. 515–527.
[85] A. Sannino, G. Postiglione, and M. H. Bollen, "Feasibility of a DC network for commercial facilities," in *Proc. 37th IAS Annu. Meet. Conf. Rec. IEEE Ind. Appl. Conf.*, 2002, pp. 1710–1717.
[86] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *J. Cryptol.*, vol. 1, no. 2, pp. 77–94, 1988.

Tianxiang Shen received the B.Eng degree from Jilin University the Excellent Engineering Class. He is currently working toward the PhD degree in Computer Science Department at the University of Hong Kong, supervised by Dr. Heming Cui. His research interests includes distributed systems, with a particular focus on the system security and data privacy.

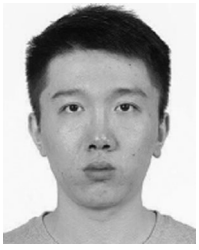
Jianyu Jiang receives the bachelor's degree from Computer Science Department at Xi'an Jiaotong University, under the supervision of Professor Qi Yong is currently working toward the PhD degree in Computer Science Department at The University of Hong Kong. He is working on topics in large scale computation platform under the supervision of Dr. Heming Cui.



Yunpeng Jiang is currently working toward the bachelor degree in information security from the Department of Computer Science and Engineering, South China University of Technology. His research interests include system security and trusted computing tecnhiques.

Xusheng Chen received the bachelor degree from the University of Hong Kong. He is currently working toward the PhD degree in Computer Science of the University of Hong Kong, under the supervision of Dr. Heming Cui. His research interests include distributed consensus protocols, distributed systems, and system security.

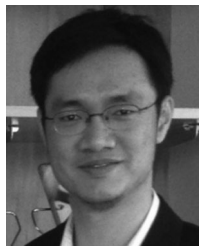
Ji Qi received the BS degree from the Beijing Institude of Technology, Beijing, China, in 2015, and the MS degree from Tsinghua University, Beijin, China, in 2018. He is currently working toward the the PhD degree in computer science at the University of Hong Kong, under the supervision of Dr. Heming Cui. His research interests include domain-specific modeling, distributed system, and cloud computing.



Shixiong Zhao received the bachelor's degree from the University of Hong Kong, and the master's degree from HKUST. He is currently working toward the PhD degree in Computer Science of the University of Hong Kong, under the supervision of Dr. Heming Cui. His research interests include distributed systems for high performance computing, distributed systems, and system security.

Fengwei Zhang is an associate professor with the Department of Computer Science and Engineering at Southern University of Science and Technology (SUSTech). His main research interests include systems security, with a focus on trustworthy execution and hardware-assisted security. Before joining SUSTech, he spent four wonderful years as an assistant professor at the Department of Computer Science at Wayne State University.

Xiapu Luo received the PhD degree from the Hong Kong Polytechnic University. He is an associate professor with the Department of Computing, the Hong Kong Polytechnic University.He spent two years at the Georgia Institute of Technology as a postdoctoral research fellow. His current research interests include network and system security, blockchain and smart contract, mobile, and IoT securit



Heming Cui (Member, IEEE) is an assistant professor in computer science of HKU. His research interests include operating systems, programming languages, distributed systems, and cloud computing, with a particular focus on building software infrastructures and tools to improve reliability and security of real-world software. Homepage: https://i.cs.hku.hk/ heming/.