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—Decentralized approaches to anonymous communication are promising to eliminate the centralized vulnerability. However, existing anonymous systems either provide limited level of anonymity under global passive attacks Traditional anonymous networks (e.g., Tor) , or use strong network synchronization to protect users, which expose large attack surface for active attackers to halt by intervening packet transmission (e. g. , Dissent). We present , a fully decentralized anonymous communication system that provides both sender-receiver unlinkability under global passive attacks , and sender/receiver anonymity under active attacks . leverages 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 paper, 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 Peer-to-Peer (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 protections of Intel SGX to regulate participants' behaviors, and ensure trustworthy message ensure trustworthy packet shuffles at each host. However, as we will show, naive processing of message shuffles within SGX still leaks sender identity to active attackers. We retain the use of DHT for efficient routing , but use randomly generated shared secret to formulate different routing circuits in each communication round, preventing active attackers distributed host, and use multiple routing paths to prevent adversaries from tracking and revealing sender/receiver identityuser identities. We show that our system is robust under machine failure and 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. All of DAENet's code and evaluation results are available on github.com/tdsc/dae-netis released on http://github.com/tdsc0652/dae-net.

1 Introduction

The Internet allows convenient communications between users, but it also incurs great anonymity concerns leads to great concerns about anonymity since communications can be censored by national security agency and even Internet Service Provider (ISPsurveilled by powerful malicious attackers such as network service providers (e.g.,chatting services), Internet Service Providers and National Security Agency (NSA). These adversaries can learn the identity of users who usually determine if two users are talking to each other by recording, tampering and analyzing communication linksanalyzing network communication traffics [1], [2], [3], [4], [5]. For example, The National Security Agency (NSA) collects user communication metadata to compromise anonymity [6]. Some countries also block anonymous services to ease censorship [7].

Formally, three kinds of attacks have been targeting at anonymous services. **1) Passive attackers** determine if two users are in a conversation by observing network packets and their metadata NSA is reported to collect internet communication (e.g.,emails and voice-over-IP chats) for crime investigations [8], and such information can be misused or leaked. Worse, some governments block targeted services (e.g.,packet size and packet rate). The most powerful attackers, Global Passive Attackers (GPA) can observe the whole network and record all network packets and their metadata. **2) Active attackers** forge, drop or duplicate network packets, and then analyze the change of network links and metadata to de-anonymize users. Senders and receivers may also collude with the active attackers. **3) Targeting DoS Attacks** completely turn down key servers of an anonymization system Telegram) that refuse to provide user communication data [7], 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.,centralized directory server/bridge in Tor) by repeatedly sending dummy messages or blocking users' accesses to centralized servers by dropping their packets. Therefore, users can only use unsafe services censored by powerful adversaries. Conventionally, **1)** and **2)** are called traffic analysis attacks Tor [9], Loopix [10]). In practice, it is desirable for an anonymous system to meet three requirements: **low-latency**, and systems against them are with strong anonymity**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 [11], [12]. Second, powerful adversaries can conduct traffic analysis by tampering, recording,

* *Contributes equally to this work.*

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 [13]. 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.

~~Existing practical anonymous networks can be classified into two categories:~~

Traditional relay-based systems ~~and shuffle-based systems. Relay-based systems~~ (e.g., Tor [9]~~and ShadowWalker [14]) forward messages through a circuit of relays, constructed randomly or statically, to hide~~ , AP3 [15]) are popular for anonymous communications. 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 ~~originator of a message. Since relays of the circuit are distributed in the whole network, simply blocking some of the relays cannot suspend the whole anonymous service, which defends against targeting DoS attacks. Unfortunately, these systems cannot defend against strong global adversaries that can observe and tamper packets transmission between relays. Specifically, they learn the originator of a message by correlating output and input of all relays, ordering the message and analyzing all the possible circuitsof the message~~relay-based approach is vulnerable to global traffic analysis attacks that can manipulate and record network packets of the relay circuits [16]. Worse, relay-based anonymous systems (e.g.,Tor) usually make use of centralized directory servers and is susceptible to targeted DoS attacks.

~~Second,~~ The emerging shuffle-based systems (e.g., Loopix [10], Dissent [17], Karaoke [18], Riposte [19], Miranda [20]~~and Yodel [21]) enable stronger anonymity guarantee by utilizing centralized servers to collect messages from users, and apply shuffles cryptographically or statistically to hide the origin of messages. To avoid malicious shuffle servers, they either use layer-based verifiable shuffles to make sure~~) 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 shuffle [22] or cryptographic shuffle [23], [24] is used. To guarantee that messages are shuffled sufficiently (i.e.,integrity), statistical shuffle assumes that the majority of ~~the centralized servers~~ machines for message shuffles are trustworthy ~~[10], [22], [25], [26]; or they apply cryptographically indistinguishable shuffles (using gamble circuit [23] and holomorphic encryption [24]) to avoid any information leakage of the the shuffle process~~ [10], [26], and cryptographic shuffle requires users to verify the integrity cryptographically [19], [27]. ~~Moreover, to~~ Second, some shuffle-based systems defend against active ~~attacks (rate controlling attacks [28]), these system typically apply synchronized messaging: each client must send exactly one message in one time interval to hide differences of message metadata (sending rate). Although these systems achieve strong guarantees against both global passive and active attacks, they can be blocked by simply dropping some or even one message [18] to the centralized servers.~~ packet drops by asking all users to send messages in synchronized rounds, such that any mis-behaved users that drop packets will be detected quickly.

~~As far as we know, none of these systems tackles targeting~~ Unfortunately, existing shuffle-based systems cannot defend against targeted DoS attacks and ~~provides strong anonymity against passive and active attacks~~ achieve low latency at the same time~~with practicality , and we owe it to two reasons. First, defending against targeting DoS attacksforbids the usage of centralized serversand calls for decentralized approaches, and~~ . 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 ~~cost-efficient to enable trustworthy~~ efficient to conduct message shuffles distributively ~~. The main reason is that guaranteeing the integrityof shuffles of one server is already costly, therefore applying them to all participants without a set~~ (i.e.,defending targeted DoS) with integrity. Both statistical and cryptographic shuffle usually make use of only a fixed, small number of centralized servers ~~produces prohibitive performance penalty.~~ to conduct shuffles efficiently.

~~Second, since the integrity of distributed participants cannot be guaranteed, compromised participants and global attackers can conduct active attacks by intervening the packets transmission process, such as controlling message rate or dropping packets [28]. Shuffle-based approaches tackle this problem by enforcing each participant to send exactly one message in one time interval (strong synchronization), but enforcing strong synchronization in a fully decentralized network is non-trivial and cannot achieve in a cost-efficient manner [29]. In fact, the original~~ However, these fixed, centralized servers are exposed to targeted DoS attacks. Specifically, statistical shuffle make messages go through a sequence of fixed servers (e.g.,own by mutually untrusted parties), and each server conduct 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 mall. However, when 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 shuffle in cryptographic shuffle can be verified, the verification cost increases exponentially on the number of shuffle nodes. For example, DC-Net [30] ~~makes use of cryptographic~~ conducts shuffles in a fully distributed manner using verifiable shuffles and all-to-all ~~broadcast distributively, which are computational and network traffic expensive. As a result, DC-Nets' latencyincreases exponentially and they typically support less than hundreds of users in~~

practicebroadcasts, 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 guarantee integrity and confidentiality of execution code and data. With confidentialityand integrity, it is potentially beneficial to anonymous messaging systemsefficiently preserve code integrity and data confidentiality [31], [32]. For example, SGX-Tor [33] hides the identifiers of hidden services and circuitsof Torby storing the entry relays of the service in SGX enclaves. SGX-Tor is the first practical work that raises the bar for network adversaries by using SGXis 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 inflations). However, SGX-Tor is still vulnerable to passive and active attacks and requires centralized directory servers to construct circuits and conduct SGX attestations. 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 practical anonymous messaging anonymous communication system based on SGX that can defend against both traffic analysis attacks and targeting DoS attacks . Building is not just simply running a distributed network in trust executing environment. First, although SGX guarantees code integrity and confidentiality, network-level adversaries can still observe network traffic outside enclaves and analyze the origin of messages. Second, participants and attackers outside enclave can arbitrarily drop or delay network packets to compromise anonymity.

To tackle passive traffic analysis attacks, we propose a meet the three desirable requirements. Specifically, all users in DAENetform an structured peer-to-peer (P2P) network with metadata (e.g.,user identifier) shielded by SGX, and makes use of SGX for trustworthy message shuffles. With the help of a structured P2P network [34], [35], DAENetcan achieve low-latency as messages need to go through only $log(N)$ users to reach the destination. Moreover, DAENetcan 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 DAENetstill needs to handle traffic analysis attacks.

First, structured P2P network has 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, DAENetproposes a *stealthy p2p network*Stealthy P2P Network to hide the identities of participants when there are no active attacks. A participant of the network joins and connects to random members of the network, and all messages are decrypted only in an SGX enclave. Therefore, outside attackers and even the participant cannot learn her location in the network. To hide network with two features. First, users in DAENetare 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 [36]. Second, to hide message patterns, we enforce our stealthy P2P network enforces trustworthy message shuffles and use a dead drop abstraction that randomly choose one node for exchanging messages from receiver and sender. Combining 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 P2P network produces oblivious packets transmission (§5) packet transmission under passive traffic analysis attacks for all participants .(§5.1)

Second, the static traffic patterns of a structured P2P network can leak 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 [37] 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.

To defend against active attacks, instead of using network synchronization, adopts a detection based design to detect malicious drop of packets. Each participant periodically sends self-loop packets to a random neighbor and routes back. After that, the participant consults each on-path relay to send back a receipt, proving that it honestly forwarded the loop packet rather than maliciously dropped the packet. This process is executed within SGX enclave, malicious participants cannot distinguish and intervene this process. Moreover, since encapsulates packets in the same format, network administrators cannot distinguish a loop packet and subvert the detection protocol.

We implement

We implemented DAENet with 5.25.2k LoC in C++ on linux. handles member admission and messaging with Linux. We use Chord [34] as the implementation of our structured P2P network, as it is an efficient and popular P2P network. DAENetproposes a membership protocol (§4.1)that attests the SGX code integrity and assists in user join. Meanwhile, DAENetproposes a dialing protocol to securely initialize conversations and exchanges the shared secret used for constructing a sequence dead drops, without leaking sensitive information to adversaries. DAENet also tolerates network churn and machine failure.

This paper makes the following contributions: failures to guarantee the liveness. We compared DAENetwith Loopix [10] and Dissent [38], two state-of-art, open-sourced shuffle-based anonymous systems. Loopix and Dissent make use of centralized servers for layer-based shuffles and cryptographic verifiable shuffles, respectively.

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

Our evaluations show that:

- This paper analyzes the potential attacks and defenses in existing anonymous messaging systemsand discuss the use of Intel SGX for improving security guarantees.
- DAENet is the first practical anonymous network that defends against both traffic analysis and targeting secure. DAENetcan defend against various attacks, including passive and active traffic analysis attacks and targeted DoS attacks. We give an end-to-end anonymity evaluation to prove that our approaches are effective.
- is efficient to scale to a DAENethas low latency when scaling up to large number of participantsusers. DAENet incurs 2.2s only 2.2s end-to-end latency with 10,000 participants. Compared with Loopix [10], a state-of-art shuffle based system, incurs 6X DAENetincurs 3X ∼ 7X higher latencythan . is robust to machine failure. After killing 30% participants, 7X lower communication latency, yet DAENetcan recover within a short time. defends against DoS attacks.

End-to-EndLatency	(s)AntiGPAAnti-RevealCircuit IDAnti-Packet drop(liveness) / (Anonymity)AntiTargeting DoS
Tor [9]	0.25 ∼ 1.5 × × √ / × ×
SGX-Tor [33]	0.525 ∼ 3.15 × √ √ / × ×
ShadowWalker [14]	> 4 × × √ / × ×
AP3 [15]	—— × × √ / × √
Loopix [10]	6.8 √ √ √ / × ×
Riposte [19]	> 3600 √ √ × / √ ×
Dissent [38]	1.3 √ √ × / × ×
Atom [39]	30.0 √ √ √ / √ ×
Karaoke [18]	6.0 √ √ × / √ ×
	2.2 √ √ √ / √ √

Comparison of to relay-based and shuffle-based systems. "√" indicates that the system can handle such vulnerability, while "×" is on the opposite. 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. §2 describes the backgroundof low-latency anonymous communication network and introduces the usage of SGX in . §3 gives an high-level overview of , presenting the roles, network topology and security primitives of introduces the security goals. §4 describes detailed 's anonymous protocols. §5 gives a security analysis. §6 is the performance evaluation. §8 introduces 7 discusses limitations and future directions. §8 is the related work , and §9 concludes our work.

2 Background

2.1 Anonymous Communication Systems

shows existing Existing anonymous communication systems ' latency and their security guarantee. These systems typically either trade security guarantees for latencyand usability (Tor), or trade latency and usability for better security properties (Karaoke) . None of them achieves strong privacy guarantees against traffic analysis and targeting DoS attacksat the same time with low latencycan be classified into two categories: relay-based systems and shuffle-based systems. Shown in Table 1, we compare DAENetto prior systems from the perspective of three requirements.

Among the relay-based systems, Tor [9] is the most popular anonymous network ever deployed, with an estimated eight million daily active users [40]. 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 which monitor the whole network and de-anonymize sender identities by correlating every input and output packets [41], [42], [43]. Meanwhile, a recent work also shows Tor's susceptibility to targeted DoS by conducting bandwidth amplification [44]. As an improved work of Tor, SGX-Tor [33] 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 [14] and AP3 [15] 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 are still observable by traffic analyzers.

To address these problems, builds from fully decentralized p2p network, which is immune to targeting DoS attacksin the first place, and we seek to provide stronger privacy guarantee by enabling both sender-receiver unlinkability and sender anonymity (§4) In contrast to relay-based systems, shuffle-based systems resist traffic analysis attacks, more precisely, passive traffic analysis attacks by means of messing up the order of input packets and output packets (i.e., shuffling). To handle active traffic analysis attacks, Riposte [19] uses Private Information Retrieval (PIR) technique to detect and stop malicious packet drops [45], [46]. 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 [39]. Loopix [10], Dissent [38] and Karaoke [18] 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 [20]. 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.

Category		Latency / Scale (#users)	Anti: Passive Traffic Analysis	Anti: Active Traffic Analysis	Anti: Targeted DoS
Relay-based	Tor [9]	0.25s ∼ 2.5s / 8M	×	×	×
	SGX-Tor [33]	0.525s ∼ 3.15s / 819	×	×	×
	ShadowWalker [14]	> 4s / 1000	×	×	√
	AP3 [15]	N/A / N/A	×	×	√
Shuffle-based	Loopix [10]	6.8s / 500	√	×	×
	Riposte [19]	> 3600s / N/A	√	√	×
	Dissent [38]	1.3s / 500	√	×	×
	Atom [39]	30.0s / 1024	√	√	×
	Karaoke [18]	6.0s / 16M	√	×	×
	DAENet	2.2s / 10,000	√	√	√

Table 1: Comparison of DAENet to existing anonymous communicating systems. "√" indicates that the system can handle such vulnerability, while " × " is on the opposite.

2.2 Structured Peer-to-Peer Network

Structured P2P network (e.g., Chord [47], Pastry [35]) 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 [14]. Also, ~~is designed to be more scalable (evaluated in 10, 000 nodes) with moderate latency (less than 2.2s end-to-end latency) compared to other state-of-art shuffle-based systems (§6).~~ a structured P2P network has the potential to hide the roles of participants by sending dummy messages along the links between every participant.

Specifically, 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 [48], ~~a structured peer-to-peer network topology as a foundation for anonymous messaging~~an efficient DHT scheme as the underlying communication protocol. In Chord, each participant ~~is assigned with an identifier when she first~~ 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 ~~within $[0, 2^b)$~~pictured as a ring which wraps modulo 2^b, and b is chosen according to the scale of network. Each participant knows only a fraction of other participants in the network. Specifically, for a participant with ~~id~~ identifier idx, ~~she~~ it is connected to b neighbor nodes who ~~has the closest id~~ have the numerically closest identifier to $idx+2^i(0 \leq i < b)$. ~~In , the neighbor nodes of a participant are called successors and the participant itself is called the predecessor of all her neighbor nodes.~~

~~Also,~~ 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 ~~KEY_j~~S, is mapped to a ~~numerically closest participant node .~~ participant node who has numerically closest identifer to S (i.e., $Map($~~KEY_j~~$S)$ = ~~$ClosestNode_{id}$~~$ClosestNode_{idx}$) by using consistent hashing [49]. 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.

~~Such topology enables efficient routing in a decentralized network. To send a message, a participant first checks if she is directly connected to the destination of the message and sends it directly if so. Otherwise, she sends the message to her furthest neighbor who is numerically closest to the destination. Chord requires only $\mathcal{O}(\log N)$ steps for a message to reach the destination, which makes it suitable for building distributed network with large amount of participants~~Although Chord facilitates efficient lookup, Chord itself does not provide anonymity guarantee 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) ~~[31], [32], [50]~~ [31], [32] is a popular security hardware available on commodity CPUs. It provides secure execution by putting data and ~~executing~~ 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 [51] to verify that a particular piece of code is running in a genuine SGX-enabled host.

Usage of SGX in . We leverage three features provided by Intel SGX to enable stronger privacy guarantee: First, we use SGX to regulate behaviors of participants in the network. SGX provides local/remote attestation service to help admit clean participants and verify the code integrity. By running 's protocol within SGX, we ensure that all participants are running the correct protocol rather than seek to misbehave. Second, we use SGX to simplify our security protocol thus achieve better performance. For example, ensures correct shuffling of network packets within SGX-enabled hosts. Hence, we do not need to use computational-heavy verifiable shuffle [22] to verify the correctness of the shuffling results. Third, we ensure confidentiality of decrypted plaintext by using SGX, so that malicious hypervisors or OS cannot see the sensitive information of network packets, such as the circuit ID.

v.s. SGX-Tor. SGX-Tor proves the feasibility of running SGX-enabled hosts to improve the security model of anonymous communication systems. SGX-Tor improves the original Tor's security guarantee, preventing 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.

leverages SGX to prevent private information leakage and regulate participants' behaviors, similar to SGX-Tor. Moreover, we further improves SGX-Tor's security model by protecting participants from network-level adversaries, such as GPAs that conduct traffic analysis attack and active attackers that maliciously drop and delay network packets. Although incurs slightly higher end-to-end communication latency compared to SGX-Tor (shown in and evidenced in §6.1), we believe that users may tolerate 's moderate latency to achieve stronger privacy guarantee in anonymous communication.

3 Overview

We begin with describing the threat model and security goals of , and introducing the roles of participants in the network that facilitate anonymous communication. Then we give the privacy approaches to enable anonymity in a fully decentralized network.

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 distinct capabilities: global observation and traffic control. allows arbitrary number of compromised participants in the network, but requires an adequate number of Confronted with such adversaries, DAENetrequires at least $k \cdot \log \mathcal{N}$ honest participants to ensure message deliveries. Specifically, requires at least $\mathcal{O}(\log \mathcal{N})$ out of $\mathcal{N}$ honest participants, in which the coefficient complete message deliveries, where the coefficient k depends on the *rounds* of communications and communication rounds of a conversation, and $\mathcal{N}$ is the total number of participants in the network. Similar to other SGX-enabled systems [33], [52], 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.

The main purpose of running anonymous protocol in a fully decentralized network is to remove the targeting DoS concern of centralized services. Moreover, to enable stronger privacy guarantee, we focus on and seek to handle the following three critical vulnerabilities under passive/active attacks.

Protecting against GPAs. Global Passive Attacks (GPAs) are proven to be the strongest passive attack in the network, that can eavesdrop on network traffic exchanged among all the participants, in order to find a message path of a communication. To determine if two participants are in a communication, GPAs may conduct prefix hijacking [53] to intercept network traffic, or off-path statistical analysis [54]. For example, in a typical traffic analysis attack, GPAs inspect the traffic within the network and keep observing the load of each participant. Based on the observation that overall traffic patterns are not particularly distorted in each hop, GPAs correlates the time of incoming and outgoing messages. Meanwhile, GPAs can learn the emitting rate of network packets at each host to distinguish a potential message sender.

Detecting eclipse attack. Eclipse attack is a major vulnerability in a p2p system. With the help of SGX, prevents adversaries from forging or duplicating network packets, including communicating messages or control messages for maintaining a structural network. However, the adversary might arbitrarily drop network packets among participants to partition a fraction of participants from the network, in order to halt the system liveness. A structural p2p network relies on nonstop sending of control messages to refresh membership and maintain its topology. If some control messages are maliciously dropped, the membership will get partitioned and communications will fail due to incomplete message deliveries. In a partitioned network, the anonymity set is smaller and the adversary will have a larger chance to de-anonymize targeting participants (§5.2).

Preventing phishing attack. As a fully asynchronous messaging system, may become the target of network adversaries who collaborate with participants and control the message emitting rate to reveal the identities of another communicator in the same conversation. In this attack, a compromised message receiver keeps holding connection and communicating with another participant in the network, while the network adversaries delay/drop packets between the sender and receiver to reveal the routing circuit, based on the observation of whether the compromised receiver has received the packet from the sender in time or not (§4.4).

Specifically, in this *phishing attack*, a malicious receiver R collaborates with active attackers to reveal the identity of sender S. Denote circuit $||C_i|| \Leftarrow < S, P^1, P^2, ..., P^{i-1}, P^i,$

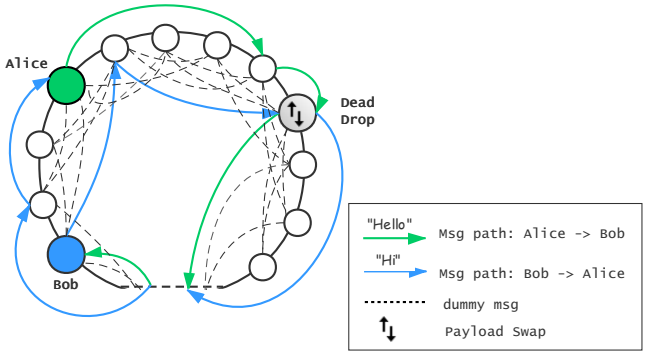


Figure 1: ~~Example~~ An example of DAENet dead drop messaging. In ~~each~~ a communication round, Alice and Bob ~~provider~~ separately sends two close-loop messages while exchanging ~~the~~ their message payload at a ~~designated~~ randomly selected dead drop node.

~~$R >$ as the routing circuit that links the malicious receiver R and victim sender S. Since the network structure is explicit to adversaries with global view (§??), R can periodically, yet slowly drops messages from her predecessors. If R drops an application message from one of her predecessor and receives no message from S in this communication round, then R can determine that this predecessor is actually P^i - the participant that acts as the previous hop in $||C_i||$ that links R and S. By repeating this process, the malicious receiver R will ultimately reveal the identity of sender S when blocking all the packets from S's predecessor does *not* effect receiving the message from S.~~

~~Note that the phishing attack applies to compromised senders as well, who collaborates with active attackers to reveal the identity of a targeting receiver.~~

~~In , all participants equally act as protocol nodes, with no designated parties, such as administrative servers or message boxes in previous anonymous communication systems [10], [38]. A participant can play multiple roles at a given time, determined by whether she is communicating or not. Therefore, targeting DoS attackers cannot distinguish a specific participant who is playing a specific role. There are totally~~

3.2 Participants As Protocol Parties

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

Relays ~~&~~ Session Nodes. Relays are idle participants. They ~~are not in any conversations and only responsible to forward packets~~ do not hold any conversations with other participants and are only responsible for forwarding messages in the network, including both application ~~packets (e.g., Alice's~~ messages (i.e., instant messages) and underlying ~~p2p control packets~~ P2P control messages (i.e., messages for maintaining ~~the~~ DAENet's structural topology).

~~**Session nodes** are in private conversations and each pair of session nodes shares a secret used for communication. To communicate with each other~~In contrast to relays, session nodes ~~use the same Pseudo Random Number Generator (PRNG [55]) with their shared secret to generate a pseudorandom sequence of keys, and these keys are mapped to a deterministic set of participants in the network. proposes a secure dialing protocol to establish a secure communication channel and negotiate communication details (e.g., the shared secret) between two session nodes (§4.2). A~~ 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 nodesexchange packet content between any pair~~ 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 (§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 (§2.2), $DeadDrop\_keys$ are deterministically mapped to a series of nodes. Hence two session nodes ~~are agreed~~ can agree on the same sequence of dead drop nodes in the network. ~~All participants can act~~ Note that all participants can be chosen as dead drop nodes~~in . Also, ,~~ and the duty of a dead drop node is ephemeral and will become invalid as soon as the dead drop ~~completes packet exchanging . (§4.4)~~ node completes payload exchanging in a particular communication round.

Figure 1 shows ~~a flow of dead drop messaging in a communication round.~~ the flow of communicating through a dead drop node in DAENet's structured P2P network. With reference to a $DeadDrop\_key$, two session nodes named Alice and Bob ~~both send their messages~~ route their messages through several relays to a designated participant (i.e., the dead drop node)~~in the network, with reference to the shared secret~~. The dead drop node waits for two application messages coming and then ~~exchanging the packet content and sending~~ exchanges the message payload and sends them back to ~~originators~~corresponding senders.

3.3 Security Goals and Defending Approaches

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

3.3.1 *Defending Against Targeted DoS Attacks*

~~**Message shuffle.**~~ **Attack Assumptions:** We consider an adversary who is determined to deny service to DAENet~~'s topology is explicit to GPAs. Adversaries can keep observing global traffic flow and deducing all the neighbors (i.e. , successors/predecessors numerically) of a node through statistical analysis (§3.3). Nevertheless,~~ network, and we make two assumptions about the capabilities and makeup of the adversary. In particular, the adversary needs not 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 [56]. Second, the adversary might avoid conduct attacks from its own network. Instead, the adversary can acquire (or rent) machines in public clusters to instantiate instances of DAENetparticipants and send dummy traffic into DAENetensures that, even with an observed topology, GPAs still cannot correlate any pairs of sender/receiver in conversations because of the trustworthy message shuffles at each network, making it hard to locate the adversary.

Defending Approach: To defend against targeted DoS attacks, DAENetparticipants 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, DAENetuses 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 de-anonymize 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 a communication, GPAs may conduct prefix hijacking [53] to intercept network traffic and then use off-path statistical analysis [54] 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 is 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.

Denote $\mathcal{N}$ as the total number of participants, the membership protocol ensures that Alice's location (i. e. , IP address) is known by only $\log \mathcal{N}$ participants that precede her (i. e. , predecessors) The shuffling process works as follows: A participant Alice maintains shuffle pools for each of its successor node. Upon receiving a network packet sent by one of Alice's predecessors, Alice first decrypts and parses the packet. If the packet is a dummy message, then Alice directly discards that packet; Otherwise, if Alice is not the destination node, she searches for the next hop by finding a numerically closest successor node in her routing table. Alice maintains a shuffle pool for each successor node (i.e. , neighbor) in her routing tableUpon receiving a message m, Alice searches for m's next hop by conducting Chord lookup. If the searched next hop next hop of m is the ith neighbor successor of Alice, then the packet message is pushed to the ith shuffle pool belongs to her belonging to Alice's ith successor.

In each protocol run, Alice pulls packets messages from each successor's *shuffle pool* shuffle pool and sends them out with a probability . If Alice does p. Given a threshold α, if p is smaller than α, Alice will not pull a packet message from the ith neighborsuccessor's shuffle pool, then she wil send . Instead, Alice encapsulates a dummy message to the with the same size as a real message, and sends the dummy message to its ith neighborsuccessor. Note that Alice may hold none packets in her messages in its ith *shuffle pool*, if so, Alice will directly encapsulate and sends shuffle pool at a particular protocol run. If that corner case happens, Alice needs not to pull messages from ith shuffle pool but will directly send a dummy message to her its ith neighborsuccessor (§4.3).

3.3.3 Defending Against Active Traffic Analysis Attacks

Round based Dead drop messaging.Attack Assumptions: Similar to other DHT based lookup schemes, 's routing is deterministic, in other word, the routing path from Alice to Bob is fixed. This poses a privacy 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 [57]. We illustrate our technique to resolve such *disclosure* attack. Also, we discuss the mitigation of other aggressive active attacks that are detectable such as traffic watermarking attacks [58] and packet hijacking attacks [59] with security analysis.

Specifically, the *disclosure* attack poses a threat to sender anonymity : an active attacker might collaborate with the message receiver and reveal sender identity by reconstructing the routing path between them. In the active 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 attack, a malicious compromised receiver holds a long-term connection with a sender, and hierarchically drop packets from the last hop to the first hop of the circuit (described in §3.3.3) to reveal the identity of the target sender.

A straw man approach is to detect such attack. However, a detection based approach is hard to conduct because (1) it's hard to verify a legal/compromised receiver and (2) the active attack is recordable. Even the conversation between

targeted participant in the network and keeps sending messages to each other. During the communication, active attackers drop messages between the sender and receiver ~~is set to expire within a fixed time, when the sender connects to the receiver again, the adversary can start from previous derivation and continue reconstructing the circuit~~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 §4.4.

Defending Approach: To defend against ~~such vulnerability~~the *disclosure* attack, DAENet~~uses a sequence~~'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*~~nodesto communicate: Dead drop nodes are virtual locations where senders and receivers deposit their messages (original packets), swap message payload and fetch messages (swapped packets) back. To initialize a conversation, two participants first negotiate a randomly generated shared secret (§4.2). The shared secret is used for generating a sequence of~~ nodes as the communication endpoints. In each communication round, two session nodes send their messages to a ~~*DeadDrop_keys. Since enables deterministic*~~ *dead drop* node and exchange corresponding messages' payload. With these ~~*KEY-ID map*~~*dead drop* ~~by building on top of Chord, thus the two participants are agreed on the same set of dead drop nodes.~~

~~Now the two participants can communicate with~~ nodes, instead of directly sending messages to each other through ~~these dead drop nodes. Communication happens in *rounds*. In round i, two participants independently send their message to dead drop node N_i with $DeadDrop\_key_i$. Each message is labeled with *session-round* pair in case of wrong payload exchanging. When N_i receives a message m, she stores it and waits for the coming of m_1 which has the same label as m. Then N_i swaps the payload of these two messages and sends them back to corresponding message originators. (§4.4)~~

a fixed circuit, two session nodes send their messages to random locations in multiple 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 on DAENet's ~~design, starting~~ anonymous communication protocol. We start from the membership protocol ~~, and presenting the shuffling strategy to defend against GPAs under consistent membership (normal case)~~that handles node join, and introduce the dialing protocol to safely initialize conversations in DAENet. Then we ~~discuss our approaches to preserve anonymity under active attacks(special case)~~present the design of the stealthy P2P network to defend against traffic analysis attacks.

4.1 Membership Protocol

~~To use the anonymous service provided by , users first join the network as participants. When node i~~ 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 ~~off-band~~ out-of-band peer discovery service (e.g., a public forum). We call that member node the ~~*joiner*~~ *guarder node*. A ~~*joiner*~~ *guarder node* serves as an attestation server to verify ~~the code integrity. If node i~~ whether an unmodified DAENet's program is executed inside a real SGX host. If the node passes the attestation, ~~then the~~ the ~~*joiner*~~ *guarder node* ~~will automatically generate an~~ replies with an automatically generated identifier, which indicates ~~node i~~the node's location in the DAENet ~~topology~~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 ~~*joiner*~~ *guarder node*. The ~~*joiner*~~ *guarder node* does a standard SGX remote attestation and succeeds with a signed report from the Intel IAS. Third, the ~~*joiner*~~ *guarder node* verifies the report, generates an ~~identity~~ identifier of node i and encrypts it with sk_i, and sends both the sealed identifier and attestation report to node i. ~~As~~ If node i passes the attestation, it ~~sends~~ will send a lookup request with its symmetric key sk_i to the ~~*joiner*~~ *guarder node* to construct its routing table. The ~~*joiner*~~ *guarder node* helps node i constructs its routing table ~~and also~~ 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 ~~*joiner*~~ *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 ~~*joiner*~~ *guarder nodes*. A newly joined node will also serve as an attestation server to verify the integrity of its ~~*joiner*~~ *guarder node*. The mutual attestation is triggered when a ~~*joiner*~~ *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 ~~*joiner*~~ *guarder node* to Intel IAS and waits for a signed report.

Note that the attestation to a ~~*joiner*~~ *guarder node* is ~~hard coded~~ hardcoded into the membership protocol and ~~enforced execution unless the~~ the execution is enforced unless the ~~*joiner*~~ *guarder node* withdraws from the network. ~~Because a malicious~~ Since SGX remote attestation can help verify the integrity of the running SGX code, if a malicious ~~*joiner*~~ *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~~, it~~. The failure of passing the SGX attestation helps the new participant take ~~action~~ actions quickly:

(1) Alert users in the ~~off-band~~ out-of-band peer discovery

service ~~,~~ to reduce the confidence of that malicious ~~*joiner*~~ *guarder node*, or immediately end up contacting with that ~~*joiner*~~ *guarder node*.

(2) Retry the admission process by switching to a new ~~*joiner*~~ *guarder node* (hopefully, one that is not malicious).

This policy limits the influence a malicious ~~*joiner*~~ *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 [60], further compromising the anonymity provided by DAENet. DAENetsolves this problem by using two approaches. First, such vulnerabilities can usually be fixed through CPU microcode updates [61], 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. DAENetrejects 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, DAENetuses 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 DAENetis anonymous file-sharing where c tries to fetch a secret file from sp_j. Since sp_j has to hide its identifer 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 a reply (i.e., time-out).

Figure 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 its 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 exists 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 in order 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 exists any initialization requests from clients. If sp_j finds any service requests, it will send 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.

Note that the existence of broker nodes for handling registration requests is not contradictory to the P2P feature of DAENetdue 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 identify of the service provider, and the service provider can securely broadcast its services and receive conversation initialization requests

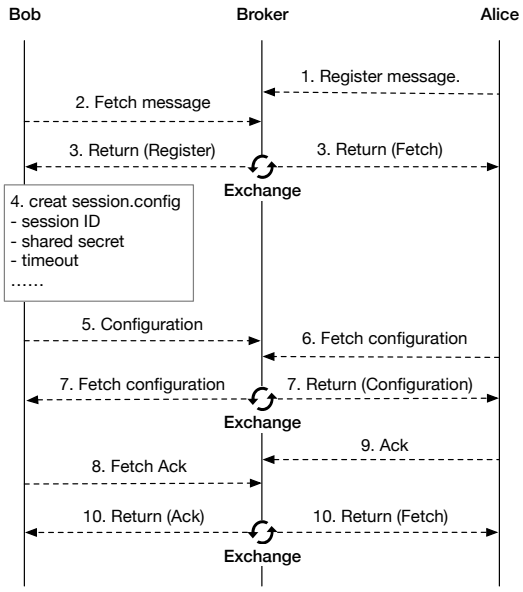


Figure 2: Two participants of DAENetinitiates their conversation through a secure dialing protocol.

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 ~~ensure the unlinkability between senders and receivers~~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, ~~GPAs cannot link another message that precedes/succeeds it~~ adversaries cannot identify its preceding or succeeding messages and further reconstruct ~~an~~ the entire routing circuit of a ~~targeting~~ conversation. We define *Sender-Receiver Unlinkability* as the inability for ~~a GPA~~ passive traffic analysis attackers to distinguish whether $\{S_{real} \rightarrow R_{real}\}$ $\{S_{real} \rightarrow R_{real}\}$ or $\{S_{real} \rightarrow R_{other}, S_{other} \rightarrow R_{real}\}$ for a real message sender ~~/receiver~~ S_{real}, a real message receiver R_{real}, and other participants S_{other}, R_{other} ~~of GPA's choice~~.

~~If a participant of sends a message immediately after receiving that message from one of her predecessors, this would allow GPAs to link these two messages and further deduce the circuit. 's~~ **Trustworthy Message Shuffling:** DAENetpreserves 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 ~~her~~ its successors. For each input message, Alice first decrypts the message by using ~~her~~ its own symmetric key, recalling that ~~for each message delivery, the~~ a sender will encrypt ~~the message by using next hop's symmetric key~~ its messages with the symmetric key of the next hop (i.e., successor). Then Alice searches for the next hop ~~by using~~ of the message with reference to the identifier of the ~~destination~~ receiver node. If the next hop for that message is the ith successor of Alice, then the message is pushed to the ~~*shuffle pool*$i$belongs to the~~ th shuffle pool belonging to Alice's ith successor.

In each protocol run, Alice totally pulls $\log \mathcal{N}$ messages from each of ~~her *shuffle pool*~~ 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*~~shuffle pool. In other ~~word~~words, the *expected shuffle rate* implies whether Alice will send an message to ~~her~~ its ith successor or not. If Alice does not pull ~~an~~ a message from the ~~*shuffle pool*~~ shuffle pool of the ith successor, then Alice encapsulates a dummy message and ~~will send that~~ sends the dummy message to ~~the~~ its ith successor. Note that ~~its~~ it's likely for Alice to hold none messages in ~~the~~ its ith ~~*shuffle pool*~~shuffle pool. In that case, Alice will directly send a dummy message to ~~her~~ its ith successor.

More precisely, when Alice receives a message x, ~~she~~ it

(1) decrypts x by using ~~her~~ 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 ~~packet~~ message header to x_id, and ~~push~~ pushes message x to ~~the shuffle poolof~~ x_id~~(i.e., *pool_id*)~~'s shuffle pool.

(3) randomly pulls l messages from ~~each successor's shuffle pool~~ shuffle pools of each successor with equal probability α.

(4) encapsulates dummy message dmy_i if Alice ~~fails to pick one~~ 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. ~~Each participant periodically, and randomly sends loop packets to detect malicious packet drop. Here, even if Host 2's query message to Host 3 is maliciously dropped, after verifying Host 3's *proof of forwarding*, Host 2 determines that Host 3 is actually online, hence maintains Host 3's membership.~~

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

~~We consider~~ **Low Attack Ability:** Consider the case where ~~GPAs~~ 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 ~~, mixes together with in~~ 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 ~~circuit of a conversation.~~

Supposing the adversary has conversation. Supposing the adversaries have a high confidence of message x being a real message (rather than a dummy message)and wants to correlate x with another real message x_1 from one circuit. The following lemma provides an upper bound on the probability that an adversary , the following claim gives an probability on which the adversaries correctly link the previously observed message x with one of the outgoing message x_1.

Theorem 2. 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

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

Note that $t+k$ is the total number of outgoing messages from Alice's host. All of the outgoing messages have 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 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 defends against traffic correlation attacks as well [10].

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 theorems claim and a security metric *likelihood* to give an end-to-end anonymity evaluation in §5of defending passive traffic analysis attacks in security analysis (§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 /output messageand output messages.

Running the shuffling protocol within SGX guarantees the integrity of shuffling operation. However, active attackers might attempt to circumvent the protocol by refusing to cooperate in message sending, result in some loss of shuffle messages. When some shuffle messages are missing, adversaries can conduct eclipse attack to ruin the networkconstruction by prevent participants from obtaining a consistent view of membership.

Although we encapsulate packet within SGX and adversaries (including the malicious host itself) cannot determine whether a send-out message is a dummy message or a real message, they can arbitrarily drop messages in the network to ruin the process of network construction, harass the membership and ultimately deny the service of .

There are totally two cases: first, if an adversary drops a dummy message or application message , the construction of the network will not be affected. The underlying p2p construction relies only on control messages, dummy messages and application messages are not involved in this process.Second, if some control messages are maliciously dropped by active attackers, the membership will get partitioned if naively uses one query message to determine whether a participant's neighbor is alive or not because a participant will remove the "offline" neighbor from its membership list even if this neighbor is actually online.

To prevent such vulnerability, 's participants periodically send *loop packets*. The sender S specifies a random bit b_s and constructs a routing path by selecting the $b_s th$ neighbor as the next destination in each hop, until encounters the sender S itself. Participants at each hop record both b_s and round number as a receipt of forwarding the loop packet. The algorithm of sending a loop packet is shown in .

$S_{id} \leftarrow$ the identity of sender S $b_s \leftarrow$ random bit generated by $Srnd \leftarrow$ the round of PACKET chosen by $Srecord(t) \leftarrow$ mark down t as a receipt of forwarding PACKET $send(p, dest) \leftarrow$ send packet p to next hop $dest$ Send_loop(PACKET) If the loop packet successfully completes its loop back, this indicates that all the participants on the routing path are alive, then the sender will send *proof of forwarding* to all the on-path participants to notify that their successors on this routing path is valid. On the contrary, if the loop packet fails to complete the loop back, this reveals that at least one of the participants on the path misbehaved. Then the sender sends receipt packets to all the on-path participants to ask for a proof of forwarding. Notice that the random bit in the loop packet is originated within SGX, even a malicious host cannot see the plaintext of that bit. Hence the random bit in loop packet is unforgeable. When receives the receipt message from the sender, each on-path participant sends back proof of forwarding independently. The sender verifies the bits in each proofs and reports the verification result back to the participants.

In , Host 2's query message to Host 3 is dropped by an adversary. Instead of removing Host 3 from her neighbor list immediately, Host 2 waits for a proof of forwarding within a fixed time interval. When Host 2 receives such proof from Alice that Host 3 forwarded her loop packet successfully, Host 2 learns that her query message may be dropped due to either unstable network environment or malicious active attackers, and will keep Host 3's membership maintained. If Host 3's proof is invalid, then Host 2 can correctly remove Host 3 from her neighbor list and re-fetch a new list. All participants that contain Host 3 will soon be notified within several protocol runs (by running Chord's topology maintenance protocol).

Note that all loop packets in are stealthy. Each participant periodically, yet randomly send loop packets to check the liveness of on-path nodes, so that adversaries cannot distinguish between a loop packet and a "regular"

packet, and drop specific loop packet to prevent sending proofs.

A special case here is when an aggressive active attacker entirely blocks the traffic from a participant to one of her successors. In that case, the loop packet will fail to complete its loop back, and the participant may consider the successor to be offline. Confronted with this attack , we argue that if a participant's query message cannot be replied for a long time, she is more likely to be monitored. To continue her conversation safely, she may rejoin the network to fetch a new list of successors.

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 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 actually 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.Approach: As discussed in §3.3, active attackers might collaborate with compromised participants to reveal the identity of target sender/receiver by means of dropping/delaying the traffic. A straw man approach is to detect malicious phishing disclosure behaviors in the network. However, detecting phishing disclosure attacks in the network is difficult and inefficient. First, naively set setting a threshold ϵ as *time-out* at sender to cut off a long-term communication is impractical because we cannot determine an average latency of communication in the network as the number of participants grows 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, communication becomes communications become hard to carry on in the network. Second, the *loop packet message detection* that is used by prior work to detect malicious packet drops does *not* work in this scenario. loop packet *Loop* message is used to prove to a participant that a potentially withdrawn neighbor is online. However, since the sender S does not know the exact or even relative position of malicious receiver R the malicious receiver in the network, loop packets *loop* messages cannot tell whether the application message drop is due to an offline receiver or malicious phishing behaviora 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 identity identifier of senders. The basic idea of this design is randomly selecting a sequence of participants in DAENet as destinations for senders/receivers two session nodes to exchange information in different 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.

Round-based Dead Drop Messaging. DAENet enforces communication communications through a sequence of *dead drops: Dead drops* dead drop nodes. Dead drop nodes are virtual locations on hosts where senders and receivers where two session nodes deposit their messages (*original packetsmessages*), swaps message payload from the same conversation and fetch messages (swapped packetsmessages) 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*. Since enables deterministic The *KEY-ID mapby building on top of Chord, thus the two participants are agreed on the same set of dead drop nodes DeadDrop_keys* are deterministically mapped to a set of nodes in the network.

Now the two participants can Two session nodes (namely Alice and Bob) communicate with each other through these dead drop nodes. Communication happens Communications happen in *rounds*. In round i, two participants independently send their message to Alice and Bob independently send a message to a dead drop node N_i with which is mapped from $DeadDrop\_key_i$. Each message is labeled with a *session-round* pair to indicate its identityunique session identity with another participant and the round of payload exchange. When N_i receives a message m, she it stores it and waits for the coming of m_1 which has the same label *session-round* pair as m. Then When m_1 arrives, N_i swaps the payload of these two messages and sends them back to corresponding message originatorssenders. The round-based dead drop messaging is effective to defend against active attacks because (1) communication circuits keep changes by setting disclosure attacks because communication circuits between session nodes changes with different dead drop nodes as destination and (2) decouples the "send-receive" relationship between two participants, thus attacks to reveal participants' identity by monitoring the messaging process (including packet drop) cannot workdestinations. 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.

To subvert the dead drop messaging design, a compromised sender might delay her message in a communication round to let the designated dead drop node in that round delay sending back the exchanged message to the receiver. Such misbehaves will not pose an anonymity concern because the adversary cannot distinguish a delayed message from **Conversation with Compromised Nodes:** Even with some fully-compromised dead drop nodes, DAENetcan still preserve anonymity due to the dead drop node to the receiver with trustworthy shuffles, so that the delayed message cannot be tracked to reveal the identity

~~of the receiver.~~ following reasons. First, adversaries cannot determine which nodes are selected as dead drop nodes in 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.

~~At the highest level, communicating through dead drops is not the first design proposed by . Prior mix-net anonymous networks (e.g., Karaoke, Vuvuzela) leverage similar idea to prevent malicious receivers from directly talking to senders and reveal senders'identities. However,~~ 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 ~~design is vulnerable, as~~ 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. In addition, in 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 DAENetscales 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, DAENettreats both cases as node failures. DAENettolerates 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 $DeadDrop\_key_j$.

DAENetprovides such flexibility because DAENetsupports 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 ~~potential to be compromised or controlled by active attackers~~randomly selected, the failure of all dead drop nodes indicates a potential monitoring of the network. Such vulnerability will be quickly detected by the session nodes, and the ~~adversaries might also observe the access pattern of dead drop nodes to determine who are in the same conversation~~session nodes are suggested to carry on their conversations later.

~~To~~

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

Trusted ~~swapping.~~Swapping: All dead drop behaviors are executed within SGX. Since SGX guarantees the confidentiality of decrypted messages in memory, ~~thus~~ a malicious dead drop node cannot determine ~~which two messages belongs~~ whether two messages belong to the same conversation and which communication pairs are being swapped.

Ephemeral ~~duty.~~Duty: ~~Dead drop nodes~~ Duties in DAENet are ephemeral which means ~~they~~ that the dead drop role do 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 current communication round, unless being chosen by the two participants again. Hence, a malicious dead drop node will not always hold the conversation and have no chance to reveal the link between the two participants§~~5~~.

~~**Unified output rate.** According to 's shuffling protocol, it's difficult for a malicious dead drop node to distinguish whether a received message is a real message or dummy message. Also, 's shuffling enforces unified output rate among honest participants to hide the access pattern of dead drop nodes. Honest participants periodically send a message (either real or dummy) to all her neighbors with a fixed output rate. Hence, if a malicious dead drop node delays/speeds up its emitting rate, she will *not* observe an expected traffic fall/spike to reveal the identity of communicating participants.~~

~~Two participants of initiates their conversation through a secure dialing protocol.~~

~~A dialing protocol helps two participants (namely, a client and a service provider) initiate their conversation in the network. However, a client needs to start her conversation without being identified, and a service provider needs a secure way to help clients reach her while preserving anonymity. This is handled by 's dialing protocol, shown in .~~

~~For the client/service provider, the whole dialing procedure proceeds within three steps through a special dead drop node, namely a *broker*. A *broker* is the service provider's designated virtual location that is responsible for receiving initialization requests in the network. A client who wants to start a conversation with the service provider dials through that *broker*, and negotiates necessary conversation configurations (e.g., the shared secret, session ID, expiry time).~~

~~With the help of *broker*, the client and service provider establish a secure channel to start their conversation. Note that in , the dialing protocol and conversation protocol execute independently and asynchronously. Since the packet size for dialing is totally the same as conversation (limited in 1KB), and the packet format is totally the same, thus the combination of dialing traffic and conversation traffic could improve 's privacy. Next, we introduce 's dialing protocol from the angel of both the client and the service provider.~~

~~To register a service of service provider, Alice (client) first finds the key of Bob from an out-of-bound channel. It could be a database where all service providers (including Bob) put their *service_key* and a *broker_key*. With the destination being the *broker_key* of Bob (KEY_Bob), Alice encapsulates a *welcome* message containing Bob's *service_key* and sends~~

~~that message by calling 's *connect()* function. As a structured p2p network, with parameter KEY_Bob, Alice's *welcome* message will finally reach a participant, which is called a *broker* in . The *broker* verifies the *service_key* to ensure that Alice wants to connect to Bob. In next round, Alice sends a fetch request to the broker to ask for the configuration file for transmission. Within the configuration file, there are totally four elements: $sessionID$, $shared\_secret$, $transmission\ rate$ and $time\_out$. Last but not least, Alice sends an ACK to the broker to invoice the arrival of the configuration file, and implies that he agrees with the *shared_secret* for secrete communication. By this step, the dialing process for a client is completed successfully.~~

~~As a service provider, Bob makes its service public by releasing its service key (Bob's public key $PubKey_{Bob}$) through an out-of-bound channel while keeping the corresponding private key $PrvKey_{Bob}$. Then, Bob registers his service to the *broker* with key $hash(PubKey_{Bob})$, using a registration message $< PubKey_{Bob}, identifier, random, Sig >$, where $Sig = Sign(identifier||random, PrvKey_{Bob})$. The *broker* first verifies the signature of this message with the public key. This verification is necessary because a user may maliciously claim that one service key belongs to him and provides the key to in order to block service of the real owner or get messages intended for the real owner. If the verification passes, the *broker* stores a mapping $< PubKey_{Bob}, identifier >$ in its enclave. When the *broker* receives a *welcome* message from any client in the network, it tries to find out the identifier of the designated service key. If succeeds, the *broker* notifies Bob for negotiating further communication details.~~

~~Through dialing, Alice now has registered itself to Bob without knowing Bob's location, Bob is aware of a service request from somewhere in the network. With a negotiated configuration file for transmission, Alice and Bob can then carry out communication.~~

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

$$x \to ... \to id \to ... \to y. \tag{3}$$

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

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

, 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 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 ~~passive traffic analysis attack~~ 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 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 ~~metric~~ 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

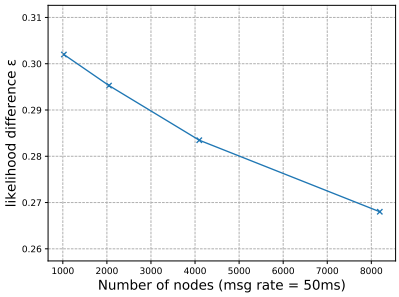


Figure 3: Likelihood difference ϵ depending on the number of participants in the network.

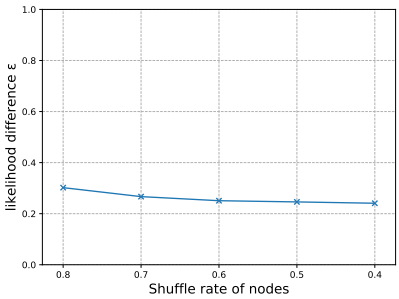


Figure 4: Likelihood difference ϵ depending on the shuffle rate for each participant in the network.

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 Equation (1) and Equation (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 unified messaging rate 50ms. Among the participants, 10% participants hold on communications while the left 90% 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% 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 (§4.3). As we pick S_0 and S_1 in their final destination, we calculate ϵ in Equation (5).

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

More participants, stronger anonymity: As we can see from Figure 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 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: Figure 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 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.

~~We assume all participants to be honest in this experiment. A fraction of compromised participants who try to delay the packets in the network have slight influence on the anonymity. Since we encapsulate all packets in the same format within trusted executing environment, a compromised participant cannot distinguish targeting messages~~ **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., ~~labeled message)and delay them to increase the *likelihood difference*~~a topology of 3 layers with 3 mix nodes per layer), when Loopix achieves comparable likelihood as DAENet(0.25), it incurs 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 §6).

5.2 Analysis of Active Traffic Analysis Attacks

In this subsection, we analyze the impact of active ~~attacks in the presence of~~ traffic analysis attacks in DAENet. First, we analyze ~~against targeting DoS attacks when network attackers compromise a fraction~~ active attacks that compromise a proportion of nodes to increase the ~~chances of participants choosing fully malicious messaging path~~chance of choosing a fully malicious routing circuit. We continue by evaluating the ~~round-based dead drop design and the security of loop messages. Likelihood difference ϵ depending on the number of participants in the network. Likelihood difference ϵ depending on the shuffle rate for each participant in the network.~~ security of anti-disclosure attack and other relevant active attacks.

~~can defend against targeting DoS~~

5.2.1 Resisting Fully Controlled Circuits

Anonymous communication systems defends against active attacks with the assumption that ~~the packets~~ 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 ~~word~~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~~ P2P lookup protocol, which is ~~correctly executed within~~ enforced to execute inside SGX, the only way the adversary can succeed in conducting ~~targeting DoS attack~~ targeted DoS attacks is by adding more compromised nodes~~. However, the adversary cannot disconnect any honest nodes from being picked in a circuit because the loop messages periodically check the liveness of honest nodes to prevent them from being withdrawn, therefore, the adversary can only~~ , in order to increase the probability of ~~being chosen as a relay in~~ choosing compromised relays in a circuit.

Denote M_{adv} as the set of compromised nodes controlled by the adversary, N as 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 ~~scale~~ scales to 10,000 participants, even with a large compromised rate ($~~p_{m1}~~ p_m$ = 0.8 ~~, p_{m2} =~~ or 0.5), the probability of successfully conducting ~~targeting~~ targeted DoS is less than 0.05 and 0.0001, respectively. ~~Even~~ 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 ~~packet sender /~~a message sender and receiver, the adversary has to ~~sabotage the entry/exit traffic and collaborate with one party to ensure~~ make sure that a conversation indeed ~~happens~~ traverses through this fully compromised circuit~~. Note that the larger the scale of , the more expensive the targeting DoS attack, and the lower the probability of success.~~ , which is hard to realize in practice.

5.2.2 Resisting Aggresive Active Attacks

~~As discussed in Section §4, active attackers can halt the anonymity through dropping and delaying packets, or even collaborate with participants to launch stronger attacks in the network. Active attackers might try other approaches to subvert the anonymity of participants. We first discuss the security of our approaches against most common and critical active attacks, and then discuss other potential~~ In this subsection, we discuss other relevant active attacks that try to de-anonymize DAENet participants.

~~A compromised participant might refuse to receive or forward predecessors' packets. If the dropped packets are application messages, such attack is imprudent because participants can simply wait for a short time and send that message in other communication round, without leaking anonymity; Otherwise, if control messages are dropped, such misbehaves will be detected by loop packets. The preceding participant can force the succeeding participant to formulate a *proof of forwarding* with a receipt to prove itself forwarded her packets.~~

~~Loop packets are secure against counterfeiting. Since the receipt must contain a secret key, which is generated by the the loop packet sender and kept secret within SGX, compromised participants and network adversaries cannot duplicate that key to counterfeit the forwarding behavior. Also, the adversary is incapable to determine a loop packet. Because all the loop packets are encapsulated in the same way as genuine messages, and processed through the shuffling. Thus, adversaries cannot seek to predict a loop packet by observing the timing pattern. As the output rate of each participant is unified, adversaries cannot determine loop packets by observing the rate of sent messages.~~

~~uses round-based dead drop design to protect senders/receivers from being observed by active attackers who collaborate with compromised participants.~~ **Defeating DAENet Anti-Disclosure Protocol:** As we discussed in §3.3, a fixed circuit in a ~~p2p~~ P2P network gives chances to attackers to hierarchically reconstruct the message path. By using the dialing protocol (§4.2) to agree on a set of dead drop nodes in the network, DAENet prohibits adversaries from tracking an honest participant and revealing ~~her identity. We~~ its identity. Also, since we can trust the PRNG used ~~in the dialing protocol~~ inside SGX to generate a series of $DeadDrop\_keys$, adversaries cannot predict the ~~next~~ every dead drop node ~~to exchange packet~~ used for exchanging message payload.

Therefore, defeating DAENet's ~~anti-phishing~~ anti-disclosure protocol requires active attackers 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 ~~phishing~~ disclosure attacker receives a delayed message whose latency is $l_1 + (T_d \times l_2)$, then ~~she~~ the attacker might have confidence to reveal the message sender.

However, precisely blocking all the on-path message is difficult ~~, meanwhile, the usable~~ and the usable attack time is short. As we will show in §6.1, the expectation of averaged end-to-end latency is less than $2.2s$. To successfully defeat the protocol, the adversary ~~has~~ 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. ~~Therefore, this attackis beyond the capability of attackers.~~ making it impractical to conduct.

Traffic Watermarking Attacks: Pointed out by Xinyuan [62], 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. DAENetcan 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 packetsmessages. For example, active attackers can launch $(n-1)$ attack [63] to track a specific message from Alice by blocking other packets messages to an honest participant. Also, network adversaries can inject malformed packets messages to replace ordinary network packetsmessages. 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 node 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 packet message dropping due to a compromised participants or network congestion. In theory, both of them can make DAENet lose its liveness while malicious packet message drop may also lead to privacy leakage (shown in §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 packets messages drops to maintain a consistent view of membership in the network, caused by either misbehaves or network congestion, thus honest participants will *not* be wrongly punished.

Proof of Randomness. requires a participant send messages to all her neighbors with the same probability. However, different sending rates can reveal sensitive messaging patterns to GPAs. Next we prove that a participant sends messages to all her neighbors with the same probability and thus achieves full randomness.

Suppose that each node in an 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.

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

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

$$x \to ... \to id \to ... \to y.$$

where 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

$$(\sum_{k=0}^{n-1-i} C_{n-1-i}^{k})(\sum_{k=0}^{i} C_i^k) = 2^{n-1-i} 2^i = 2^{n-1}$$

, which is identical to all id's neighbors.

's protocol is carefully managed to reveal only a small, yet insensitive set of variables to adversaries. First, by running code inside SGX, we prevent adversaries from directly intervening the protocol execution and seeing the decrypted plaintext of network packets. Malicious participants can monitor traffic links and deduce a set of her predecessors/successors under 's topology. However, she cannot distinguish whether a received packet from a predecessor is a control message - used to maintain the network structure, or an application message - used for communication.

Second, '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 packets 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 unify the output rate of participants, observing the output rate does not reveal any sensitive information as well.

To conclude, reveals three insensitive observable variables to adversaries: closest neighborhood (i.e., predecessors/successors of a participant), number of sent-out packets in each round, and the output rate. This set is significantly smaller than previous anonymous messaging systems, enabling to minimize the useful information exposed to adversaries in the network.

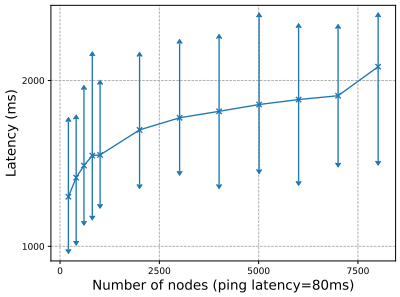


Figure 5: Latency of DAENet when 50 to 8000 participants simultaneously send traffic at rate $\mu = 50ms$ and shuffle messages with probability $\delta = 0.8$. We assume that there is no additional delay add by participants.

6 Evaluation

Our evaluation was conducted on 20 computers with SGX-equipped Intel(R) Xeon(R) CPU E3-1280 v6 with 24 cores, 64GB RAM and 2TB SSD. All computers form a cluster with 40Gbps network. We used to set intra or inter machines

~~network latency~~ In our cluster, each machine runs multiple (up to 400) instances of DAENetclient. We used Linux Traffic Control (`TC`) to set the network latency between clients as $40ms$ to simulate ~~Internet network environment. Each machines runs multiple instances of the~~ Internet environment.

~~We built a chatting application for evaluating the performance . The chatting clients connect to a server for communications and will send a message to server when it receives the server's reply or a timeout is triggered. The chatting server simply replies to all messages received from clients. We sampled 10% of all participants as clients/servers, and other participants will work as normal relays. Except for the node scalability evaluation, we use 1,000 participants to evaluates the performance. We set the timeout of message in chatting as 10s.~~

We compared DAENet~~with two~~ 's performance with two state-of-art shuffle-based ~~systems (Loopix and Dissent) which have an approximative security guarantee. Loopix is an~~ anonymous communication systems: Loopix and Dissent. Loopix is a popular open-sourced ~~, shuffle-based anonymous network defending against traffic analysis. We ran 7 mix servers, 10 providers in the clusters and various number of clients in Loopix. We sampled 10~~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 DAENetand has been proved in §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 [64], 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 DAENetand 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% of all ~~Loopix clients to serve as a server or a client of our applications. We wrote an interface to connect the two application (chatting and file Sharing) to Loopix clients~~participants as active message senders while other participants still work as normal relays. The ratio X% is set to 10% by default, with reference to the Daily Active Users (DAU) of the popular WhatsApp application [65]. As Loopix ~~hides the sender's identifier and does not support message reply, we attached the sender id in each message for message reply. We also compared with Dissent, an open-sourced~~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,~~shuffle-based anonymous network. We wrote an interface to forward traffic of the Chatting application to Dissent and ran a Dissent private clusterfor evaluations.~~ 000 clients) to evaluate the performance.

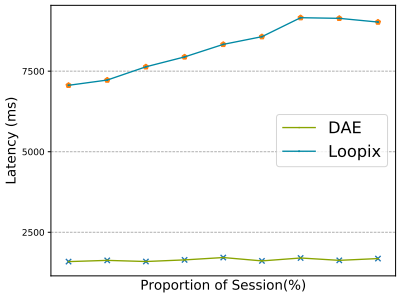


Figure 6: Latency of DAENet for anonymous communicating for varying number of sessions. The latency does *not* increases as the number of session grows.

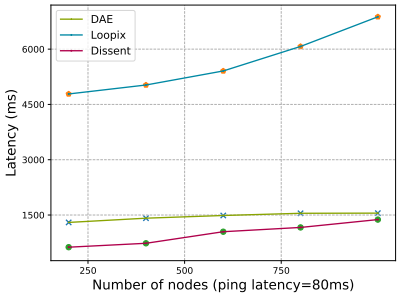


Figure 7: Latency comparison. We measured Loopix and Dissent - two state-of-art scalable anonymous messaging systems.

~~Quantitatively, our~~ Our evaluation answers the following ~~research~~ questions:

- §6.1 Can DAENet support a large number of participants and provide acceptable performance?
- §6.2 How sensitive is DAENet to its parameters?
- §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 ~~research~~ questions in this subsection:

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

~~Horizontally~~ 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 Figure 5, the latency increases logarithmically with increasing number of users. When 8000 participants send traffic simultaneously the latency is nearly 2000ms.

Note that the latency overhead increases logarithmically with total number of participants. This is because the underlying topology of DAENet is a structured ~~p2p~~ 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

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

Table 2: Bandwidth cost of running DAENet.

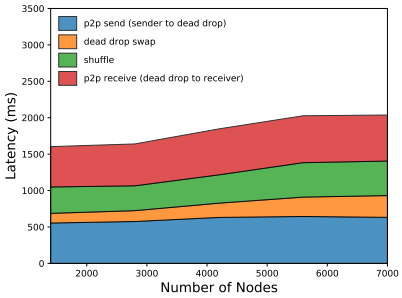


Figure 8: Breakdown of DAENet latency.

DAENet scales to 1M participants, the expected path length for a lookup only grows to 20.

Number of messages. ~~To evaluates~~ : To evaluate how the number of ~~communication session~~ active nodes affects latency, we ~~increases~~ increased the proportion of active nodes (i.e., nodes in communication sessions) from 10% to 95%, and measured the network latency, as shown in Figure 6. As the proportion of active ~~nodes~~ node increases, DAENet's latency does not increase much, while Loopix's increases dramatically. This is because Loopix ~~incurred large shuffle overhead while the number of sessions increases, while~~ incurs larger shuffle overhead with a growing number of messages through its centralized mix servers. On the contrary, participants in DAENet~~can identifies them~~still send dummy messages even if there is 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.

~~Breakdown of DAENet latency.~~

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 ~~simulate~~ 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 ~~systems that claimed~~ system that claims to be scalable to users. We picked the system parameters $\mu = 50ms$ 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.

Figure 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 the 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

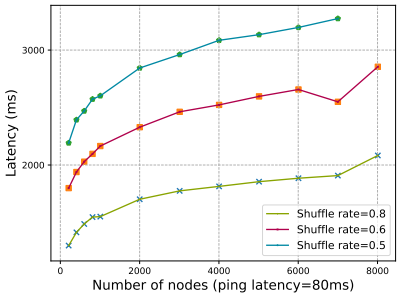


Figure 9: The end-to-end latency of DAENetwith unified message emitting rate 50ms and varying shuffle rate.

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

Table 3: The end-to-end latency of DAENetwith varying message emitting rate, running in a cluster of 1000 nodes.

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~~ P2P communications, as shown in Figure 8. Around 69.1% of the latency is from ~~p2p~~ P2P communication, as it requires $log(N)$ steps to locate a node in the network. Dead drop communication contributes 8.4% of the latency. The last source of the latency, message shuffling, incurs only 22.5% of the latency.

As we can see from the breakdown results, DAENet will slow down the communication by adding 30.9% more round-trip latency. However, we believe that DAENet is useful for anonymous online communications, as participants may value 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 ~~only~~ has one conversation ~~with randomly picked sender/receiver~~ between two randomly picked participants from all instances.

To understand the minor bandwidth cost (around 0.14MB/s), DAENet's design crucially avoids heavy usage of network resource for sending dummy messages. This is because we also add ~~p2p~~ P2P control messages to the shuffle pools, so that when participants have to send out a dummy message to a neighbor, ~~she~~ it can just replace by sending a control message rather than a ~~useless~~ dummy message.

~~The cost is independent of the number of participants. With more participants, the number of application messages increases with more conversations in the network, so that the bandwidth cost will not rise rapidly.~~ The sending of control messages in DAENetfollows 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 Figure 9 and Table 3. With unified message emitting rate $50ms$, 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 larger emitting rate. However, the descending trend of latency is smoother with large emitting rate. This is because that the latency is also bounded by dead drop swap and ~~p2p~~ P2P communication.

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

~~Message emit rate (μs)100 500 1000 2000 30005000Latency (ms) 773 758 787 10551302 1594~~

~~The end-to-end latency of with varying message emitting rate, running in a cluster of 1000 nodes.~~

~~Arbitrarily killing nodes to simulate network churn with a 10% killing rate. Arbitrarily killing Loopix nodes.~~

6.3 Failure Recovery

Handling node churn is a major issue in ~~p2p~~ P2P systems. To evaluate the failure resilience of DAENet, we ran DAENet for a period of time, with a typical message emitting rate $50ms$ and shuffle rate 0.8, and we arbitrarily killed 10% of all participants for three times (totally killed 30% active participants). ~~To compare the robustness, we also run Loopix to its limit (800 nodes) and arbitrarily kill Loopix nodes as . The killed nodes of Loopix includes at least one mix nodes to evaluate the failures of centralized servers.~~ The killed nodes of DAENet are sampled uniformly from existing participants, including both active participants (communicating) and idle participants.

Figure 10 ~~and~~ shows the latency before and after killing nodes. When nodes are killed, ~~both~~ DAENet~~and Loopix~~'s latency becomes extremely high because the lost of transferring message triggers timeout. After that, DAENet's latency resumes to normal in a short time, as DAENet detects failure of message, updates routing table and resumes processing. ~~On the other hand, the performance of Loopix after killing is unstable. This is because client nodes are unknown about the failure. In this way, the latency is reduced only when a client node chooses a working server. With more failed server, the Loopix system triggers more timeout and communications become hard to carry on, which may not be tolerated by some users.~~

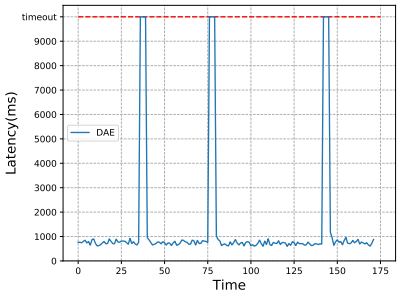


Figure 10: Arbitrarily killing DAENet nodes to simulate network churn with a 10% killing rate.

7 Discussion

DAENethas two limitations. First, current DAENetimplementation do 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 [66], [67], it is fixable by using well-known Oblivious Ram (ORAM) algorithms, such as ZeroTrace [68].

Second, DAENetcurrently 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 DAENetfrom 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 [9] is the most popular onion routing system. Due to its popularity and transparent development processes [69], many researchers have explored attacks that can de-anonymize Tor users and hidden-service providers by monitoring the network traffic. Recent attack vectors for Tor include BGP-based attacks [70], [71], website fingerprinting [72], [73], [74], [75], traffic correlation [42], [43], [76], [77], congestion attack [78], [79] and targeted DoS [44], [80], [81]. Meanwhile, researchers also propose methods to enhance Tor's security by optimizing the bandwidth report for selecting guard nodes [82] and monitoring circuit construction [83]. 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 [84], [85].

TEE and SGX-Tor: TEE provides strong security guarantees (i.e., confidentiality and integrity) for applications with efficiency. Intel SGX [86] 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 [52], [87], network analysis [88] and secure key-value stores [89]. SGX-Tor [90] is the first work that applies SGX to

anonymous network. ~~Unfortunately, it only defends against specific Tor adversaries and protects circuit /hidden service identifers without protecting users from~~ 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 such as global passive attackers and active attackers, potentially preventing it from being a choice of users who value strong privacy.

~~DC-Net [30], [91], [92], [93] leaks limited sensitive information by using expensive cryptographic primitives. Although it provides stronger security guarantee, it typically incurs large bandwidth consumption and higher latency, making it only suitable for applying in a small scale. To reduce the large latency incurred by DC-Nets, recent work (Pung [27], Atom [39] and Dissent [17]) introduces centralized servers to avoid expensive computation and network bandwidth. However, the centralized serversmake DC-Nets vulnerable to~~ **DAENet v.s. SGX-Tor:** DAENetalso 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 DAENetincurs 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 guarantee in anonymous communication.

Comparisons to Other Mix Networks: Vuvuzela [94] is secure against passive traffic analysis attacks. Vuvuzela's insight is to minimize the sensitive observable variables to adversaries with differential privacy techniques. By adding noise messages and mixing with real messages, adversaries cannot distinguish which users are communicating. Vuvuzela requires all messages pass through a single chain of mix servers, making it susceptible to targeted DoS attacks. In contrast, DAENet~~can defend against both traffic analysis~~ does not require a set of centralized mix servers, all messages are shuffled through each hop inside SGX. Moreover, DAENetoffers fault-tolerance to node leaving or failures, guaranteeing the liveness of anonymous communication.

Loopix [10] uses *cover traffic* and ~~DoS attacks in the same time, while incurring low latency~~*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 use *loop* messages to detect such attacks. However, Loopix cannot detect 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 DAENetis secure against all these attacks.

~~Mix-nets [10], [95], [96], [97], [98], [99] shuffle network traffic in centralized servers to hide identities of senders and receivers. Although it is easy to implement Mix-nets, most Mix-nets cannot defend statistical traffic analysis, and their formal worse-case guarantees are usually weak [100]. Vuvuzela [94], Stadium [101] and~~ Miranda [20] 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. DAENetruns in an asynchronous network so that it does not need secure clock synchronization protocol which is costly. DAENetcan 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 [17] is based on DC-networks [102]. 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 [18] ~~adopt differential privacy to defend against traffic analysis~~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 millions users which is the largest evaluation scale to our best knowledge. However, ~~all of them cannot defend against active attacksor DoS attacks [103], [104], [105], [106], [107], [108]. Karaoke even stops a whole communication round when one packet is dropped. Another~~ 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. DAENetshuffle messages through a group of trustworthy shuffling nodes in a fully decentralized network and provides fault-tolerance to DoS attacks. (2) Karaoke requires users 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, DAENethandles the initialization and hides metadata during the dialing process.

Alternative Approaches: There are two approaches in literature that have the potential to be used to enable verifiable shuffling operations in mix networks. The first approach is to ~~block all the mis-behaved participants to achieve anonymity, which is also proved to be practical in some cases [109]~~use zero-knowledge proofs [110] to verify that the mix servers have correctly shuffled messages. The second approach is randomized partial checking (RPC) pointed out in the Miranda paper [20]. 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 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~~is~~, the first work that enables strong anonymity in a fully decentralized network. DAENet~~provides both sender-receiver unlinkability under global passive attacks, and sender/receiver anonymity under active attacks. is built upon a structural p2p network for efficiency and adopts sophisticated techniques, including the dead drop abstraction and shuffling for defending traffic analysis .~~ 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~~makes use of Intel SGX to protect the integrity and confidentiality of the structural network and message shuffles.~~ forbids disclosure attacks that track and reveal sender identifers. We evaluated the latency and bandwidth cost of DAENet, and our evaluation results show that DAENet~~'s evaluation shows that it~~ scales well with moderate end-to-end latency while maintaining constant-size bandwidth requirements for users.

REFERENCES

[1] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on tor by realistic adversaries," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 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, uS Patent 9,105,300.

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

[4] R. Jansen, M. Traudt, and N. Hopper, "Privacy-preserving dynamic learning of tor network traffic," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 1944–1961.

[5] R. W. Lai, K.-F. Cheung, S. S. Chow, and A. M. So, "Another look at anonymous communication," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[6] E. Stoycheff, "Under surveillance: Examining facebooks spiral of silence effects in the wake of nsa internet monitoring," *Journalism & Mass Communication Quarterly*, vol. 93, no. 2, pp. 296–311, 2016.

[7] P. Winter and S. Lindskog, *How the great firewall of china is blocking tor*. USENIX-The Advanced Computing Systems Association, 2012.

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

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

[10] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The loopix anonymity system," in *26th USENIX Security Symposium USENIX Security 17)*, 2017, pp. 1199–1216.

[11] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking 2002*, vol. 4673. International Society for Optics and Photonics, 2001, pp. 156–170.

[12] M. T. Alam, H. Li, and A. Patidar, "Bitcoin for smart trading in smart grid," in *The 21st IEEE International Workshop on Local and Metropolitan Area Networks*, 2015.

[13] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of tor," in *2005 IEEE Symposium on Security and Privacy (S&P'05)*. IEEE, 2005, pp. 183–195.

[14] P. Mittal and N. Borisov, "Shadowwalker: peer-to-peer anonymous communication using redundant structured topologies," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 161–172.

[15] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach, "AP3: Cooperative, decentralized anonymous communication," in *Proceedings of the 11th workshop on ACM SIGOPS European workshop*. ACM, 2004, p. 30.

[16] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of tor," in *2005 IEEE Symposium on Security and Privacy (S&P'05)*. IEEE, 2005, pp. 183–195.

[17] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale," in *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation OSDI 12)*, 2012, pp. 179–182.

[18] D. Lazar, Y. Gilad, and N. Zeldovich, "Karaoke: Distributed private messaging immune to passive traffic analysis," in *13th USENIX Symposium on Operating Systems Design and Implementation OSDI 18)*, 2018, pp. 711–725.

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

[20] H. Leibowitz, A. M. Piotrowska, G. Danezis, and A. Herzberg, "No right to remain silent: isolating malicious mixes," in *28th USENIX Security Symposium USENIX Security 19)*, 2019, pp. 1841–1858.

[21] D. Lazar, Y. Gilad, and N. Zeldovich, "Yodel: strong metadata security for voice calls," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. ACM, 2019, pp. 211–224.

[22] U. Parampalli, K. Ramchen, and V. Teague, "Efficiently shuffling in public," in *International Workshop on Public Key Cryptography*. Springer, 2012, pp. 431–448.

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

[24] P. Paillier *et al.*, "Public-key cryptosystems based on composite degree residuosity classes," in *Eurocrypt*, vol. 99. Springer, 1999, pp. 223–238.

[25] D. Lazar, Y. Gilad, and N. Zeldovich, "Karaoke: Distributed private messaging immune to passive traffic analysis," in *13th USENIX Symposium on Operating Systems Design and Implementation OSDI 18)*, 2018, pp. 711–725.

[26] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM, 2015, pp. 137–152.

[27] S. Angel and S. Setty, "Unobservable communication over fully untrusted infrastructure," in *12th USENIX Symposium on Operating Systems Design and Implementation OSDI 16)*, 2016, pp. 551–569.

[28] T. Swart and H. Ferreira, "Insertion/deletion correcting coding schemes based on convolution coding," *Electronics Letters*, vol. 38, no. 16, pp. 871–873, 2002.

[29] M. Iansiti and K. R. Lakhani, "The truth about blockchain," *Harvard Business Review*, vol. 95, no. 1, pp. 118–127, 2017.

[30] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of cryptology*, vol. 1, no. 1, pp. 65–75, 1988.

[31] G. Noubir and A. Sanatinia, "Trusted code execution on untrusted platforms using intel sgx," *Virus bulletin*, 2016.

[32] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, "Malware guard extension: Using sgx to conceal cache attacks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2017, pp. 3–24.

[33] 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 Transactions on Networking*, vol. 26, no. 5, pp. 2174–2187, 2018.

[34] 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.

[35] J. Song and S. Wang, "The pastry algorithm based on dht." *Computer and Information Science*, vol. 2, no. 4, pp. 153–157, 2009.

[36] F. de Asís López-Fuentes, I. Eugui-De-Alba, and O. M. Ortíz-Ruiz, "Evaluating p2p networks against eclipse attacks," *Procedia Technology*, vol. 3, pp. 61–68, 2012.

[37] K. Park and H. Lee, "On the effectiveness of probabilistic packet marking for ip traceback under denial of service attack," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 1. IEEE, 2001, pp. 338–347.

[38] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale," in *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation OSDI 12)*, 2012, pp. 179–182.

[39] A. Kwon, H. Corrigan-Gibbs, S. Devadas, and B. Ford, "Atom: Horizontally scaling strong anonymity," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 406–422.

[40] A. Mani, T. Wilson-Brown, R. Jansen, A. Johnson, and M. Sherr, "Understanding tor usage with privacy-preserving measurement," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 175–187.

[41] M. AlSabah and I. Goldberg, "Performance and security improvements for tor: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, pp. 1–36, 2016.

[42] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on tor by realistic adversaries," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 337–348.

[43] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A new cell counter based attack against tor," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 578–589.

[44] R. Jansen, T. Vaidya, and M. Sherr, "Point break: A study of bandwidth denial-of-service attacks against tor," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1823–1840.

[45] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 1995, pp. 41–50.

[46] R. Ostrovsky and V. Shoup, "Private information storage," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 294–303.

[47] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 1, pp. 17–32, 2003.

[48] 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 Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[49] T. Schutt, F. Schintke, and A. Reinefeld, "Structured overlay without consistent hashing: Empirical results," in *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, vol. 2. IEEE, 2006, pp. 8–8.

[50] M. Orenbach, P. Lifshits, M. Minkin, and M. Silberstein, "Eleos: Exitless os services for sgx enclaves," in *Proceedings of the Twelfth European Conference on Computer Systems*. ACM, 2017, pp. 238–253.

[51] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for cpu based attestation and sealing," in *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, vol. 13. ACM New York, NY, USA, 2013.

[52] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "VC3: Trustworthy data analytics in the cloud using sgx," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 38–54.

[53] H. Ballani, P. Francis, and X. Zhang, "A study of prefix hijacking and interception in the internet," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 265–276, 2007.

[54] Y. Gilad and A. Herzberg, "Spying in the dark: Tcp and tor traffic analysis," in *International symposium on privacy enhancing technologies symposium*. Springer, 2012, pp. 100–119.

[55] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 3–30, 1998.

[56] R. Kapelko, "Towards fault-tolerant chord p2p system: analysis of some replication strategies," in *Asia-Pacific Web Conference*. Springer, 2013, pp. 686–696.

[57] D. Agrawal and D. Kesdogan, "Measuring anonymity: The disclosure attack," *IEEE Security & privacy*, vol. 1, no. 6, pp. 27–34, 2003.

[58] "Adding watermarks to images using alpha channels," http://php.net/manual/en/image.examples-watermark.php.

[59] G. Danezis and L. Sassaman, "Heartbeat traffic to counter (n-1) attacks: red-green-black mixes," in *Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, 2003, pp. 89–93.

[60] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the intel sgx kingdom with transient out-of-order execution," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 991–1008.

[61] "L1 terminal fault," https://software.intel.com/security-software-guidance/software-guidance/l1-terminal-fault.

[62] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in *2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 2007, pp. 116–130.

[63] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: Active attacks on several mix types," in *International Workshop on Information Hiding*. Springer, 2002, pp. 36–52.

[64] D. Kedogan, D. Agrawal, and S. Penz, "Limits of anonymity in open environments," in *International Workshop on Information Hiding*. Springer, 2002, pp. 53–69.

[65] C. Montag, K. Błaszkiewicz, R. Sariyska, B. Lachmann, I. Andone, B. Trendafilov, M. Eibes, and A. Markowetz, "Smartphone usage in the 21st century: who is active on whatsapp?" *BMC research notes*, vol. 8, no. 1, p. 331, 2015.

[66] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi, "Software grand exposure:sgx cache attacks are practical," in *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[67] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache attacks on intel sgx," in *Proceedings of the 10th European Workshop on Systems Security*, 2017, pp. 1–6.

[68] S. Sasy, S. Gorbunov, and C. W. Fletcher, "Zerotrace: Oblivious memory primitives from intel sgx." *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 549, 2017.

[69] T. B. Tracker, "Wiki," 2018.

[70] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal, "{RAPTOR}: Routing attacks on privacy in tor," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 271–286.

[71] Y. Sun, A. Edmundson, N. Feamster, M. Chiang, and P. Mittal, "Counter-raptor: Safeguarding tor against active routing attacks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 977–992.

[72] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 605–616.

[73] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 1187–1203.

[74] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, 2009, pp. 31–42.

[75] S. Li, H. Guo, and N. Hopper, "Measuring information leakage in website fingerprinting attacks and defenses," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1977–1992.

[76] Z. Ling, J. Luo, W. Yu, X. Fu, W. Jia, and W. Zhao, "Protocol-level attacks against tor," *Computer Networks*, vol. 57, no. 4, pp. 869–886, 2013.
[77] S. J. Murdoch and P. Zieliński, "Sampled traffic analysis by internet-exchange-level adversaries," in *International workshop on privacy enhancing technologies*. Springer, 2007, pp. 167–183.
[78] N. S. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on tor using long paths." in *USENIX Security Symposium*, 2009, pp. 33–50.
[79] J. Geddes, R. Jansen, and N. Hopper, "How low can you go: Balancing performance with anonymity in tor," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2013, pp. 164–184.
[80] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann, "The sniper attack: Anonymously deanonymizing and disabling the tor network," Office of Naval Research Arlington VA, Tech. Rep., 2014.
[81] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz, "Denial of service or denial of security?" in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 92–102.
[82] R. Snader and N. Borisov, "A tune-up for tor: Improving security and performance in the tor network." in *NDSS*, vol. 8, 2008, p. 127.
[83] A. Panchenko, F. Lanze, and T. Engel, "Improving performance and anonymity in the tor network," in *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2012, pp. 1–10.
[84] M. Soltani, S. Najafi, and R. Jalili, "Mid-defense: Mitigating protocol-level attacks in tor using indistinguishability obfuscation," in *2014 11th International ISC Conference on Information Security and Cryptology*. IEEE, 2014, pp. 214–219.
[85] M. Juárez, M. Imani, M. Perry, C. Dıaz, and M. Wright, "Wtf-pad: toward an efficient website fingerprinting defense for tor," *CoRR, abs/1512.00524*, 2015.
[86] Intel, "Software guard extensions programming reference."
[87] 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 *NSDI*, 2017, pp. 283–298.
[88] M.-W. Shih, M. Kumar, T. Kim, and A. Gavrilovska, "S-nfv: Securing nfv states by using sgx," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 2016, pp. 45–48.
[89] C. Priebe, K. Vaswani, and M. Costa, "Enclavedb: A secure database using sgx," in *EnclaveDB: A Secure Database using SGX*. IEEE, 2018, p. 0.
[90] 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 *NSDI*, 2017, pp. 145–161.
[91] D. Mills, "Dcnet internet clock service," Tech. Rep., 1981.
[92] M. F. Iacchetti, G. D. Marques, and R. Perini, "Operation and design issues of a doubly fed induction generator stator connected to a dc net by a diode rectifier," *IET Electric power applications*, vol. 8, no. 8, pp. 310–319, 2014.
[93] M. Ranjram and P. W. Lehn, "A multiport power-flow controller for dc transmission grids," *IEEE Transactions on Power Delivery*, vol. 31, no. 1, pp. 389–396, 2015.
[94] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM, 2015, pp. 137–152.
[95] P. Golle and A. Juels, "Parallel mixing," in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 220–226.
[96] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type iii anonymous remailer protocol," in *2003 Symposium on Security and Privacy, 2003.* IEEE, 2003, pp. 2–15.
[97] S. Davies and A. Moore, "Mix-nets: Factored mixtures of gaussians in bayesian networks with mixed continuous and discrete variables," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000, pp. 168–175.
[98] R. Küsters, T. Truderung, and A. Vogt, "Formal analysis of chaumian mix nets with randomized partial checking," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 343–358.
[99] M.-C. Silaghi, "Zero-knowledge proofs for mix-nets of secret shares and a version of elgamal with modular homomorphism." *IACR Cryptology ePrint Archive*, vol. 2005, p. 79, 2005.
[100] C. Diaz and A. Serjantov, "Generalising mixes," in *International Workshop on Privacy Enhancing Technologies*. Springer, 2003, pp. 18–31.
[101] N. Tyagi, Y. Gilad, D. Leung, M. Zaharia, and N. Zeldovich, "Stadium: A distributed metadata-private messaging system," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 423–440.
[102] A. Sannino, G. Postiglione, and M. H. Bollen, "Feasibility of a dc network for commercial facilities," in *Conference Record of the 2002 IEEE Industry Applications Conference. 37th IAS Annual Meeting (Cat. No. 02CH37344)*, vol. 3. IEEE, 2002, pp. 1710–1717.
[103] K. Park and H. Lee, "On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets," in *ACM SIGCOMM computer communication review*, vol. 31, no. 4. ACM, 2001, pp. 15–26.
[104] H. Zhang, P. Cheng, L. Shi, and J. Chen, "Optimal dos attack scheduling in wireless networked control system," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 843–852, 2015.
[105] H. Wang, L. Xu, and G. Gu, "Floodguard: A dos attack prevention extension in software-defined networks," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2015, pp. 239–250.
[106] J. E. Belissent, "Method and apparatus for preventing a denial of service (dos) attack by selectively throttling tcp/ip requests," Sep. 7 2004, uS Patent 6,789,203.
[107] Y. Zhang, Z. M. Mao, and J. Wang, "Low-rate tcp-targeted dos attack disrupts internet routing." in *NDSS*. Citeseer, 2007.
[108] H. Liu, "A new form of dos attack in a cloud and its avoidance mechanism," in *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*. ACM, 2010, pp. 65–76.
[109] P. P. Tsang, A. Kapadia, C. Cornelius, and S. W. Smith, "Nymble: Blocking misbehaving users in anonymizing networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 2, pp. 256–269, 2009.
[110] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of cryptology*, vol. 1, no. 2, pp. 77–94, 1988.

Tianxiang Shen is a PhD student supervised by Dr. Heming Cui in Computer Science Department at the University of Hong Kong. His research interests lie in distributed systems, with a particular focus on the system security and data privacy. Prior to his current program, he received B.Eng degree from JiLin University the Excellent Engineering Class.

Jianyu Jiang is currently a third year PhD student 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. Prior to his current program, Jianyu receive his Bachelor's Degree in Computer Science Department at Xi'an Jiaotong University, under the supervision of Professor Qi Yong.

Yunpeng Jiang is currently pursuing his Bachelor degree in information security at Department of Computer Science and Engineering, South China University of Technology. His research interests includes system security and trusted computing tecnhiques.

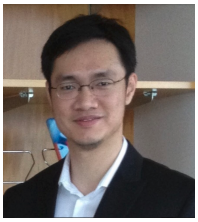
Xusheng Chen received his Bachelor degree in HKU. He is currently a PhD student in Computer Science of HKU (2017-now). He is under the supervision of Dr. Heming Cui. His research interests include distributed consensus protocols, distributed systems and system security.

Ji Qi received his B.S (2015) degree from Beijing Institude of Technology, Beijing, China, and his M.S (2018) degree from Tsinghua University, Beijin, China. He is currently pursuing the PhD in computer science at the University of Hong Kong under the supervision of Dr. Heming Cui. His interests include domain-specific modeling, distributed system and cloud computing.

Shixiong Zhao received his Bachelor degree in HKU and his master degree in HKUST. He is currently a PhD student in Computer Science of HKU (2017-now). He is 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 at Department of Computer Science and Engineering at Southern University of Science and Technology (SUSTech). His primary research interests are in the areas of 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 Department of Computer Science at Wayne State University.

Xiapu Luo is an associate professor at Department of Computing, The Hong Kong Polytechnic University. He obtained his Ph.D. degree from the same university and then 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 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/. He is a member of IEEE.