Security and Safety, Vol. 4, 2024024 (2025)

Security and Safety in Network Simulation and Evaluation

Research Article OPEN ACCESS

Information Network

K8s-enhanced lightweight simulation method for the Tor network

Wentao Huang[1], Han Wu[1], Zeyu Li[1], Xiaqing Xie[1], Qingfeng Zhang[2], Xuebin Wang[2], Jiapeng Zhao[1], and Jinqiao Shi[1,*]

[1] *Beijing University of Posts and Telecommunications, Beijing 102206, China*
[2] *Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100085, China*

Received: 8 September 2024 / Revised: 25 October 2024 / Accepted: 31 December 2024 / Published online: 5 March 2025

Abstract Criminals exploit the robust anonymity afforded by Tor for illicit purposes, prompting heightened interest among researchers in de-anonymization attacks on the Tor network. The execution of experiments on de-anonymization attacks within a real Tor network presents considerable challenges, hence the necessity for a simulation environment. However, existing methods for simulating the Tor network are inadequate regarding realism, flexibility, and scalability, with some being prohibitively expensive. In this paper, we develop a lightweight and scalable Tor network simulation environment based on Kubernetes (K8s), employing Docker containers to simulate Tor relays. The results demonstrate that a network of up to a thousand Tor relays can be simulated using just four standard hosts. Furthermore, two de-anonymization attack experiments were conducted within this simulated environment, which exhibited high levels of realism and flexibility. Finally, a hybrid networking approach combining multi-granularity relays was explored to enhance further the balance between realism and cost in Tor network simulations.

Keywords Tor, Network simulation, De-anonymization attacks

Citation Huang W, Wu H and Li Z et al. K8s-enhanced lightweight simulation method for the Tor network. Security and Safety 2025; **4**: 2024024. https://doi.org/10.1051/sands/2024024

1 Introduction

The Second Generation Onion Router (Tor) [1] is the current leading low-latency anonymous communication system, boasting thousands of volunteer relays and millions of users globally [2]. It achieves anonymity by routing traffic through multiple hops, effectively concealing the IP addresses of clients and specific servers. Additionally, Tor ensures the integrity and confidentiality of message data through layered encryption. This robust anonymity, while beneficial for legitimate users, also enables criminals to engage in illicit activities securely. For example, criminals build illegal trading services and anti-social propaganda services in the Tor network, and hackers use the Tor network to carry out anonymous ransom attacks and other hacking attacks [3–8].

To investigate criminal behavior through de-anonymization attacks on the Tor network, researchers must conduct a significant number of experiments. These experiments require the deployment of controlled relays, the customization of Tor relays, and the gathering of extensive intermediate data, including logs and traffic information [9–20]. However, conducting experiments within the actual Tor network presents several challenges. First, the deployment of a considerable number of relays is costly and logistically challenging, as Tor officials restrict this practice to prevent any single organization from compromising the network's overall anonymity. Secondly, the wide distribution of relays in the actual Tor network makes it difficult to obtain comprehensive data, which hinders researchers from observing the anonymous communication processes of Tor users in a complete manner. Furthermore, ethical concerns emerge, as experiments could inadvertently undermine user anonymity and impact the network's performance [21]. Consequently, a simulation approach to the Tor network is essential, providing researchers with a controlled environment to address the cost, complexity, and ethical issues inherent in real-world experiments.

The existing methods for simulating the Tor network can be classified into three principal categories: generic discrete event simulators, software-based simulations, and hardware-based emulations [22].

- ***Generic Discrete Event Simulators:*** These approaches utilize a general-purpose simulator to simulate discrete events within the Tor network. However, since they do not run the actual Tor application, they lack the requisite realism, thereby limiting their practical application.
- ***Software-Based Simulations:*** This category includes a variety of tools that have been developed with the specific purpose of facilitating Tor research. These tools are capable of recreating realistic network topologies and simulating large-scale relays. Nevertheless, these approaches typically read and write memory rather than transmitting actual traffic and restrict the real-time operation of Tor relays during simulations. This results in a deficiency in both realism and flexibility. Furthermore, some software-based approaches encounter difficulties with scalability.
- ***Hardware-Based Simulations:*** These approaches employ clusters of virtual hosts or hardware devices, such as Raspberry Pis, to construct dedicated Tor networks. While they offer a high degree of realism and flexibility, their cost can be prohibitively high. Overall, each simulation method presents unique strengths and limitations, highlighting the need for a balanced approach that maximizes realism and scalability while minimizing costs.

This paper introduces a novel approach to simulate the Tor network and establishes a verification platform for experimental testing. The approach employs container technology [23] to create isolated environments for Tor applications, including both network and file systems. Additionally, this approach uses Kubernetes (K8s) cluster technology [24] to manage multiple hosts, facilitating distributed simulations effectively. This approach balances the realism, flexibility, scalability, and cost reduction of Tor network simulation. To further reduce the cost, this paper further discusses the Tor network simulation based on hybrid networking of multi-granularity relays, where some relays are guaranteed to be more realistic, while others sacrifice some of their realism to reduce the resource cost of simulating the Tor network.

This paper contributes as follows:

- We propose a novel Tor network simulation method that utilizes Docker as the fundamental unit, enabling lightweight, flexible, and realistic simulations.
- With K8s, we establish a scalable experimentation platform capable of simulating a private Tor network with up to a thousand relays using just four standard hosts.
- Within the Tor network experiment platform built in this paper, we conduct a circuit association experiment and a hidden service traceability experiment, both designed to validate the realism and flexibility of the simulated Tor network.
- We further discuss a Tor network simulation method based on hybrid networking of multi-granularity relays to further balance realism and cost.

The remainder of the paper is organized as follows. Section 2 presents background knowledge and related work. Section 3 presents the Tor network simulation method of this paper. Section 4 experimentally verifies the realism, flexibility, scalability, and low cost of the Tor network simulation method in this paper. Section 5 discusses the Tor network simulation method based on hybrid networking of multi-granularity nodes. It also discusses the limitations of the Tor network simulation method in this paper and presents conclusions.

2 Background

This section begins with an overview of the Tor network, followed by a description of de-anonymization attacks and the necessity of simulating the Tor network. It concludes with an overview of related work in Tor network simulation and a summary of the research objectives of this paper.

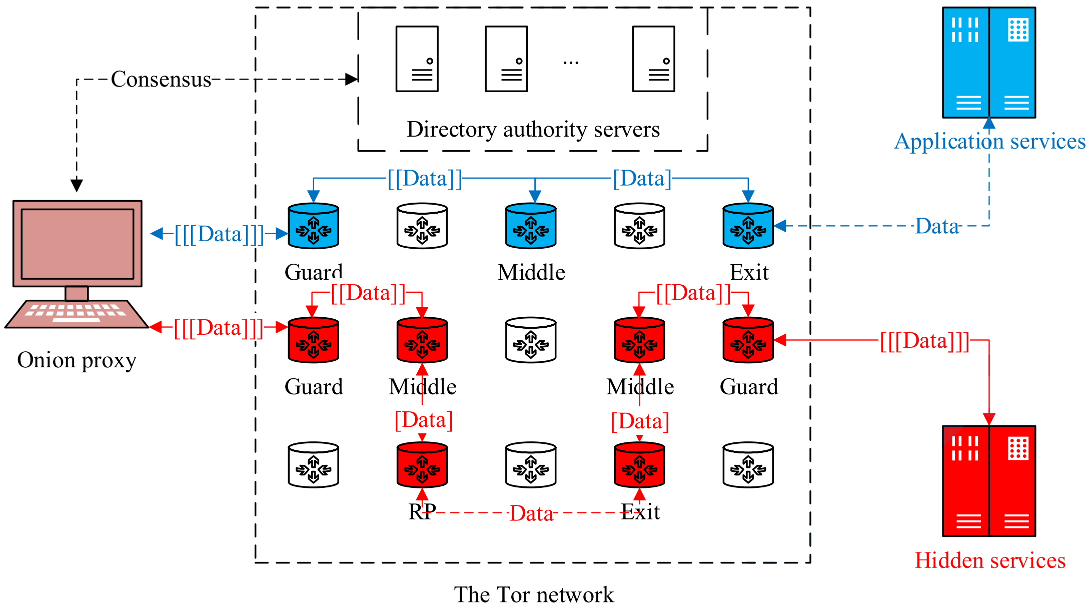

Figure 1. Diagram of the Tor network structure, where blue indicates access to common application services, and red indicates access to hidden services in Tor

2.1 Tor

The Tor network is designed to provide users with a secure and privacy-protected environment for Internet communication, as depicted in Figure 1. Users can access websites anonymously via the Tor browser [25]. Alternatively, Tor can be used as an anonymous proxy for applications such as SSH and FTP. The main components of the Tor network include:

- **Directory Authority Servers:** These central servers support the Tor network's operation and have hard-coded IP addresses within the Tor code. Relays must upload their descriptor information to these servers to join the network. The Directory Authority Servers then vote to include the relay in a consensus file, which documents details such as the relay's IP address, fingerprint, bandwidth, port, Tor version, and identity tag.
- **Onion proxy:** This user-oriented client is integrated into the Tor browser or functions as a proxy for accessing the Tor network.
- **Guard:** Identified as the first hop relay in a Tor circuit. The "Guard" relay cannot be manually designated. Instead, it must meet stringent criteria related to stability and uptime. This requirement makes deploying Guard relays costly in the real Tor network. De-anonymization experiments often require the deployment of Guard relays to capture data traffic.
- **Middle:** An ordinary routing relay that holds the middle position in a Tor circuit.
- **Exit:** The final hop relay in a Tor circuit is responsible for directing traffic to the intended destination.
- **RP (Rendezvous Point):** This relay facilitates communication between the Onion proxy and hidden services, serving as the last hop in a circuit randomly chosen by the Onion proxy.
- **Hidden service (Hidden server):** A specialized service is accessible only through the Tor network, which can host various applications, including SSH, web services, and FTP.

When a user accesses the Tor network through an Onion proxy, the proxy first requests the consensus file from the Directory Authority Servers. With the information in this file, the Onion proxy applies a path selection algorithm to choose relays for building circuits. For typical application service access, the Onion proxy establishes a three-hop circuit to connect to the target service, negotiating a symmetric key with each relay along the circuit. This process results in the data being encrypted three times by the Onion proxy before transmission and decrypted once at each relay as it travels to the final destination. The Exit relay is the sole entity capable of decrypting the innermost layer, thereby acquiring the real data. However, it only knows the address of the Middle relay and cannot identify the original sender. The Guard relay only knows the sender's address, while the address of the accessed service and the real data remain hidden. The Middle relay is unaware of the sender's address, the address of the accessed service, and the real data. This ensures the anonymity of the data sender.

Upon accessing a hidden service, a user is randomly assigned a circuit by the Onion proxy, which then designates the last-hop relay as the Rendezvous Point (RP) relay. Subsequently, the Onion proxy notifies the target hidden service of the RP relay's address via alternative circuits and procedures [26]. Thereafter, the hidden service establishes a circuit to the Onion proxy's RP relay, thereby initiating communication between both parties. In this configuration, the anonymity of both the user (the Onion proxy) and the hidden service is preserved, as the three-hop circuit and multi-layer encryption anonymize the identity of the data sender.

2.2 De-anonymization attacks against the Tor network

The anonymity provided by Tor is increasingly exploited by criminals, who utilize it to establish illicit hidden services for the anonymous trading of firearms and drugs, as well as to conduct hacking attacks [3–8, 27]. In response, researchers develop de-anonymization attacks on Tor to trace and identify these individuals. These attacks can be categorized based on their focus: user-side de-anonymization attacks and hidden service-side de-anonymization attacks [28].

User-side de-anonymization attacks aim to uncover the connection between a user and a target service, mainly using traffic correlation techniques [9, 10, 12–14, 29–32]. In this attack scenario, the attacker initially deploys several controlled relays within the Tor network. Upon detecting malicious traffic on the target service or controlled Exit relays, the attacker correlates this with the traffic on the controlled Guard relays. If successful, they can trace the user's (onion proxy) IP address. For example, Nasr *et al.* [10] use a CNN model to correlate exit relay-side and user-side Tor traffic. Guan *et al.* [29] propose ResTor, a preprocessing model for removing network noise to improve flow correlation accuracy. Tian *et al.* [30] defend against flow correlation attacks by constructing generalized perturbations to traffic. Oh *et al.* [12] further improve the correlation precision by traffic slicing and metric learning [33]. Hafeez [31] *et al.* use GPUs to accelerate real-world flow correlation attacks. Guan *et al.* [32] propose a Blind Analyzer based on pattern reconstruction techniques to remove perturbations from flows and improve correlation accuracy. Lopes *et al.* [14] propose SUMo, which correlates users with hidden services.

Hidden service-side de-anonymization attacks focus on tracing the hidden service. This requires the deployment of controlled Guard relays, and we can classify the attacks as passive or active based on the type of attack. Passive attacks primarily use the WFP (Web Fingerprinting) techniques [15–17, 34, 35]. In the initial phase, the attacker gathers the traffic of the target hidden service to train the recognition model. Subsequently, during the detection phase, the attacker employs the trained model in controlled Guard relays to identify any target hidden service traffic in the passing traffic, ultimately tracing its IP address. For example, Kwon *et al.* [36] fingerprint hidden services by observing the circuit communication pattern between the client and the hidden service. Hayes *et al.* [37] propose a k-fingerprinting technique to provide more robust website fingerprinting with the help of communication traffic. Panchenko *et al.* [38] propose a two-phase approach to fingerprint hidden services using traffic, where phase 1 filters out the traffic for hidden services and phase 2 fingerprints the specific hidden services. Henri *et al.* [39] split the communication traffic over multiple networks to defend against website fingerprinting attacks. Xu *et al.* [40] propose Zoomer, which extracts traffic features that are more useful for website fingerprinting. Wang *et al.* [41] further address the lack of training data when fingerprinting hidden services. Active attacks primarily use the watermarking method [18–20, 42, 43], whereby the attacker creates a watermark by injecting, modifying, and deleting traffic and transmits the watermark to the target hidden service via the onion proxy. If the attacker detects the watermark on the controlled Guard relays, the IP address of the hidden service can be traced. For example, Chen *et al.* [44] de-anonymize hidden services by Rendezvous cookies and circuit watermarks. Lavazza *et al.* [45] exploit Tor's congestion control mechanism for embedding watermarks to de-anonymize hidden services. Chen *et al.* [18] de-anonymize hidden services by modulating latency signals at controlled Guard relays. Zhang *et al.* [46] de-anonymize hidden services by constructing watermarks during Tor protocol handshakes.

Conducting experiments on de-anonymization attacks within the actual Tor network presents several challenges. Firstly, a successful de-anonymization attack depends on the target user or hidden service utilizing controlled relays deployed by the attacker. The extensive number of relays in the Tor network necessitates deploying a significant number of controlled relays, particularly Guard relays. However, the deployment is costly and complex for researchers. Secondly, it is challenging to ensure that the target user or hidden service consistently utilizes relays controlled by the attacker, given that Tor periodically alters its circuit paths [47]. This variability makes it hard to observe complete user behavior within the actual Tor network. Furthermore, the execution of a de-anonymization attack within the real Tor network may potentially compromise the anonymity of ordinary users, thereby raising ethical concerns.

It is necessary to establish a private and controlled network environment using a Tor network simulation to address the above challenges. However, the effectiveness of de-anonymization attacks in the simulation environment depends on meeting the following conditions.

- ***Realism:*** The simulation must utilize actual Tor applications and generate genuine Tor traffic. This is crucial, as de-anonymization attacks require the analysis of real Tor protocol packets and involve multiple complex phases.
- ***Scalability:*** The simulation environment needs to be sufficiently large and scalable. Smaller network sizes hinder researchers' ability to assess the likelihood of a target user connecting to a controlled relay during a de-anonymization attack. Moreover, with the Tor network expanding from over 6000 relays in 2022 to nearly 8000 in 2024 [2], the simulation should be adaptable to accommodate larger networks in the future.
- ***Flexibility:*** The simulation environment should enable the real-time operation of Tor relays. Since de-anonymization attacks involve multiple phases, each requiring different traffic processing and script execution strategies, the ability to operate Tor relays in real-time can significantly aid researchers.

2.3 Related work and motivation

Existing methods for simulating the Tor network fall into three main categories: generic discrete event simulator-based simulations, software-based simulations, and hardware-based simulations. Generic discrete event simulator approaches typically employ tools such as OMNET++ and NS2 to create network simulations. For example, in 2009, O'Gorman *et al.* [48] used the Java SSFNet simulator to model correlation attacks in the Tor network. In 2013, Doswell *et al.* [49] utilized OMNET++ to model the construction times of Tor circuits for mobile users at varying walking speeds. In 2022, Musa *et al.* [50] classified and evaluated the discrete event simulator in more detail. It should be noted that general-purpose network simulators are not specifically designed for Tor and are unable to run real Tor applications or generate real Tor traffic. These tools tend to be used only in the preliminary stages of research.

Researchers have developed a range of software-based simulators for the Tor network. In 2011, Jansen *et al.* introduced Shadow [51], a simulator that encapsulates real Tor applications and loads them into memory simultaneously. Shadow intercepts and redirects socket connections, event libraries, and function calls from the aforementioned applications to the simulation environment. While it is capable of simulating sizable and varied Tor networks, its scalability is constrained by its reliance on a single host for specific events. That same year, Bauer *et al.* developed ExperimenTor [52], a tool constructed on the ModelNet network simulation testbed [53]. It requires at least two hosts: one for simulating network configurations (*e.g.*, bandwidth, topology) and the other for running the actual Tor application. Wacek *et al.* [54] used ExperimenTor to analyze which relay selection technique provides the best anonymity and performance. AlSabah *et al.* [55] conducted experiments on a real Tor network and supplemented their study with ExperimenTor simulations. In 2012, Elahi *et al.* introduced COGS [56] as a means of investigating the impact of the selection of Tor Guard relays on user privacy. The tool employs the consensus file to construct a model of the extant Tor network, conducts comprehensive path selection simulations, and generates logs for subsequent analysis. In 2013, Johnson *et al.* proposed the Tor path simulator TorPS [57], which aims to simulate the relay selection process during the establishment of Tor circuits and is used to study the improvement or modification of path selection algorithms in Tor. Singh's SNEAC [58], which was introduced in 2014, is based on a modified version of Mininet and Open vSwitch. SNEAC is similar to ExperimenTor in that it requires at least two hosts, one for simulating the network and the routing system and the other for simulating specific instances. Jansen *et al.* continued to enhance Shadow from 2014 to 2022 [59–61], introducing improvements such as more precise geographic modeling of the Tor network, runtime optimizations, realistic TCP connection limits, and enhancements to the network stack, along with increased simulation speeds. However, these modifications did not address the inherent scalability issues. In 2020, Cheng *et al.* [62] used the shadow simulation environment to evaluate the improved path selection algorithm on the performance of the Tor network. The software-based Tor network simulation methods under discussion tend to be lacking in realism and flexibility. This is due to their dependence on memory reads and writes rather than actual traffic transfers, as well as their inability to adapt to highly dynamic conditions.

Hardware-based simulation methods for the Tor network often rely on high-performance servers that run clusters of virtual machines or utilize devices like Raspberry Pis to construct private Tor networks. One prominent platform is PlanetLab [63], a globally distributed overlay network that enables researchers to conduct experiments on designated segments, known as "slices", within the network. With 645 locations and 1335 nodes, PlanetLab serves as a representative model of the Internet. Researchers such as Bauer *et al.* [64] and Akhoondi *et al.* [65] have employed the PlanetLab platform to simulate and examine the routing algorithms utilized by Tor. Another platform, designated as DeterLab [66, 67], has been developed to serve as a cybersecurity simulation testbed. In contrast to PlanetLab, DeterLab enables the configuration of network topologies and the execution of arbitrary applications. For example, Chakravarty *et al.* [68] conducted a traffic analysis attack against Tor users using DeterLab, introducing bandwidth fluctuations on the server side and monitoring the resulting changes within the circuit. In addition, Emulab [69] implements a network simulation testbed through a virtual machine where researchers can provide network topologies that can then be used to generate simulated networks. Das *et al.* [70] used Emulab to analyze the impact of selective DoS attacks on user anonymity. In 2018, Wang *et al.* [71] used the OpenStack platform to build a private Tor network. In 2020, Wang *et al.* [22] created a private Tor network consisting of 20 nodes using Raspberry Pi and verified its performance through experimentation. This approach offers advantages over other similar platforms, including low cost, ease of scalability, and effective recovery of network protocols. In 2022, Zheng *et al.* [72] proposed LUNAR to build a simulation platform using virtualization and software-defined networking technologies.

A significant limitation of hardware-based simulation techniques for the Tor network is that they are constrained to simulate a finite number of relays that are equal to the available virtual machines or hardware devices. This approach is often costly and is not optimal for lightweight simulations with limited resources. Similarly, generic discrete event simulator-based methods are inadequate for conducting de-anonymization attack experiments since they do not involve real Tor applications or generate real Tor traffic. Existing software-based simulations also fall short in terms of realism, flexibility, and scalability for these types of attacks. Therefore, this research seeks to develop a lightweight, flexible, realistic, and scalable simulation method for the Tor network to effectively support de-anonymization attack experiments.

3 Method

This section outlines the design and deployment process of our K8s-based Tor network simulation approach.

3.1 Design

To address the challenges of achieving lightweight, realistic, and flexible simulations, we employ Docker container technology. To further enhance scalability, we incorporate Kubernetes (K8s) cluster technology. The overall design of the Tor network simulation is depicted in Figure 2. Within the Docker image repository, we maintain a variety of images for entities such as Directory Authority Servers, relays, and Exit relays. The images above are deployed as containers during the simulation. Moreover, users have the flexibility to store customized images in the repository, which may include, for example, FTP and web server images. The *kube-apiserver* component serves as the management interface for the entire simulated private Tor network. In this configuration, Pods serve as the primary scheduling units in Kubernetes, executing the Docker containers that represent the diverse entities of the Tor network. The master node coordinates the distribution of Pods across multiple physical hosts, utilizing actual IP addresses. Additionally, it can assign simulated IP addresses to each Pod via a network plugin, ensuring uninterrupted communication among Tor relays from disparate node hosts within the private network.

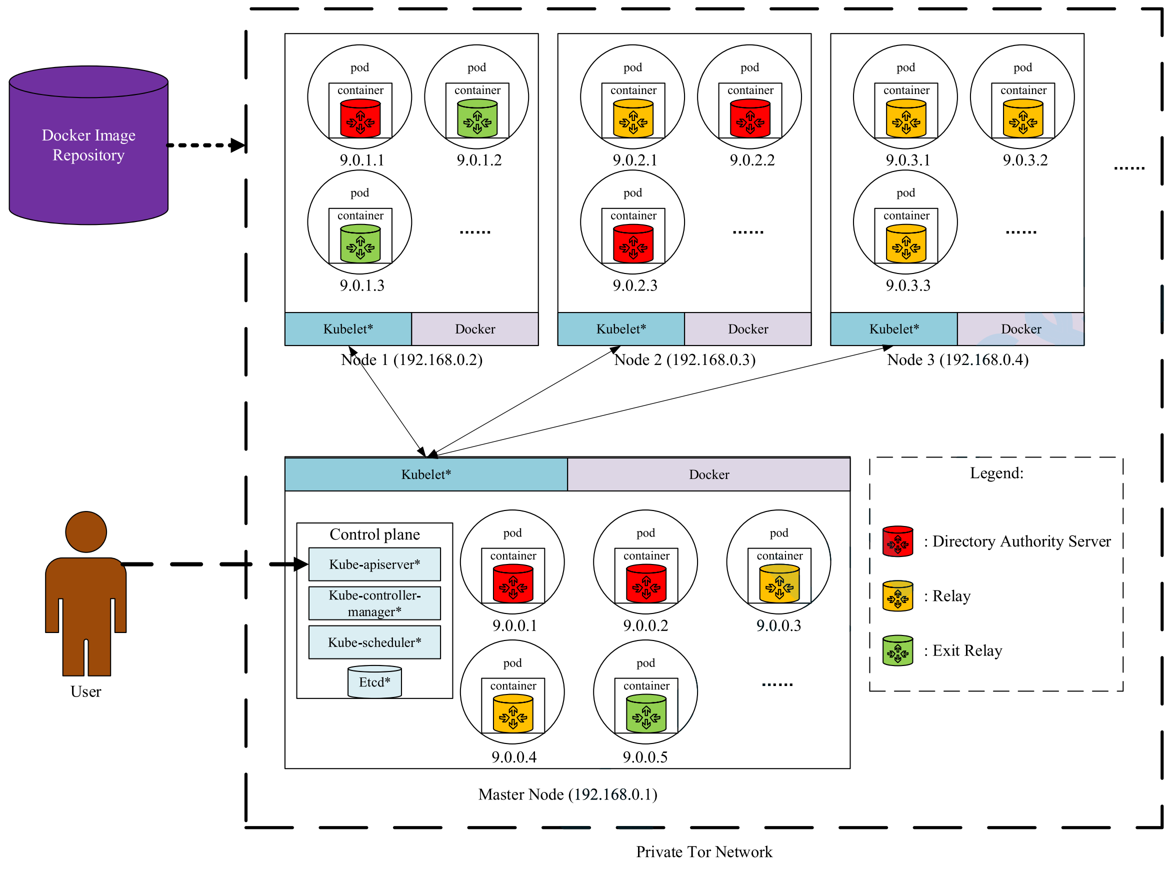

Figure 2. Tor network simulation environment based on K8s. * represents the components of K8s, of which *etcd* is the database that stores the state of the entire cluster; *apiserver* provides the only entry point for users to operate resources; *controller-manager* is responsible for maintaining the state of the cluster; *scheduler* is responsible for scheduling Pods to the appropriate machines according to a predefined scheduling policy; *kubelet* is responsible for maintaining the container lifecycle, as well as Volume (CVI) and network (CNI) management

The use of Docker technology is necessary. In order to ensure a lightweight, realistic, and flexible simulation environment for the Tor network, this paper employs Docker to establish isolated settings for each application process. In contrast to virtualization technology [73], which relies on a hypervisor to virtualize hardware components such as the CPU and memory and necessitates installing a guest operating system on this virtualized hardware, Docker operates differently. As depicted in 3, Docker does not virtualize hardware or operating systems. Instead, it provides isolation for application processes by configuring various namespace parameters via the Docker engine. This enables the processes to share the host's CPU, memory, and other resources, resulting in a reduction in overhead compared to traditional virtual machines. The architectural design of this simulation is based on software, which makes it suitable for lightweight scenarios. Docker helps to isolate distinct networks, file systems, and other necessary resources for each application process. Within this network setup, each application processes within its isolated environment, comprising a distinct network, file system, and other necessary resources. Researchers can access and manipulate application processes within the Docker containers in real-time, thereby introducing an additional degree of flexibility to the environment. In order to ensure scalability, this paper integrates Kubernetes (K8s) clustering technology. This guarantees that Docker containers across different node hosts can communicate effectively, thus enabling the seamless expansion of the network by adding more node hosts.

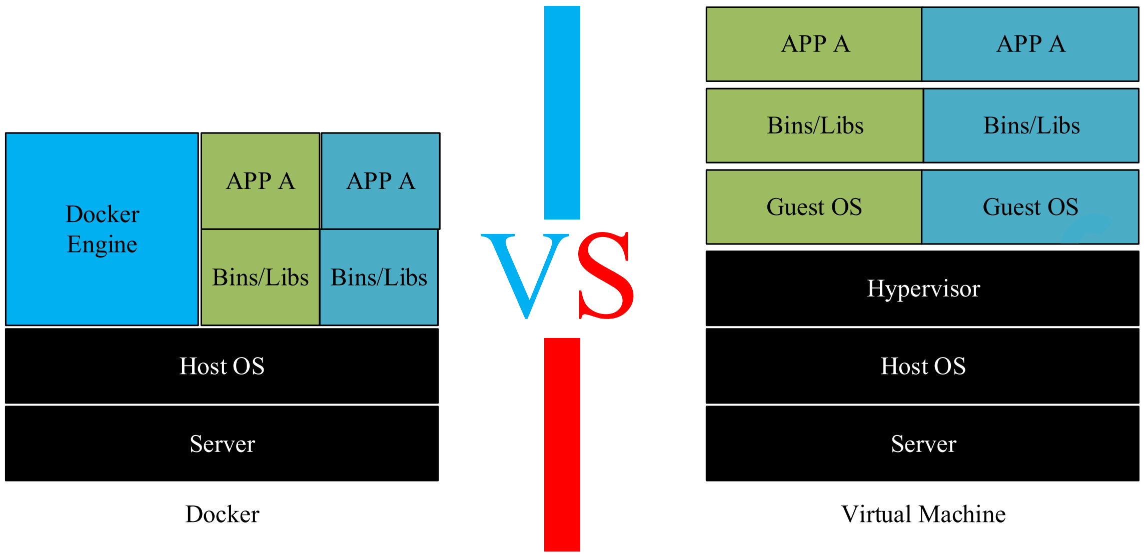

Figure 3. Docker technology vs. virtualization technology

3.2 Deployment process

3.2.1 Creating docker images

In this paper, we initially construct Tor relay images using the *Dockerfile* [74] to establish Tor relays within a simulated private Tor network. Tor relays can be classified according to their role in the network, including Directory Authority Servers, common relays, and Exit relays, among others. These Tor relays operate the same Tor application but utilize distinct *torrc* configuration files at startup. To create a Directory Authority Server, one must append *AuthoritativeDirectory 1* and *V3AuthoritativeDirectory 1* to the *torrc* configuration file. Similarly, to establish an Exit relay, one must add *ExitRelay 1* to the *torrc* configuration file. For additional configuration options, please refer to the official Tor document [75]. We also give specific configuration examples and guidance on the subsequent deployment process[1].

3.2.2 K8s cluster build

This paper employs the Kubernetes (K8s) system to create a clustered environment to simulate a private Tor network. The process begins with the deployment of a master node, along with additional nodes and the requisite network plugins. Firstly, we use the *kubeadm init* command [76] to configure the master node, which launches the control panel components as containers. The *kubelet* component must be installed separately on the host. The master node is primarily responsible for managing and scheduling resources within the system and executing user commands when necessary. Once the master node has been established, we use the *kubeadm join* command to add further nodes where pods can be deployed. It is also possible to allocate pods to the master node. In order to facilitate network communication, this paper employs the *flannel* [77] network plugin, which handles IP address allocation through the *etcd* component, distributing distinct IP address segments to each node. For example, the networks of the three node hosts in the cluster are assigned the subnets *9.0.1.0/24*, *9.0.2.0/24*, and *9.0.3.0/24*, respectively.

3.2.3 Build a private Tor network

This paper ultimately illustrates the simulation of a private Tor network, following the workflow depicted in Figure 4. The primary idea is to use private Directory Authority Servers to establish this network. In the initial phase, we deploy private Directory Authority Servers to generate identity data, including IP addresses and fingerprints, which are then documented in a publicly accessible file. Subsequently, the configuration files of other Tor relays are updated to include these private Directory Authority Servers

[1] https://github.com/mr00huang/TorSimulation.git

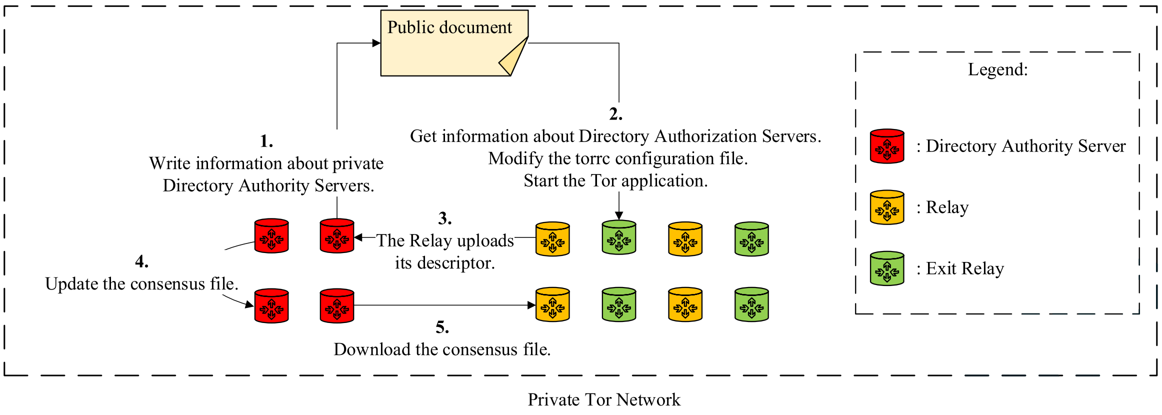
Public document
1.
Write information about private
Directory Authority Servers.
2.
Get information about Directory Authorization Servers.
Modify the torrc configuration file.
Start the Tor application.
3.
The Relay uploads
its descriptor.
4.
Update the consensus file.
5.
Download the consensus file.
Legend:
: Directory Authority Server
: Relay
: Exit Relay
Private Tor Network

Figure 4. Private Tor network organization process

by referencing the aforementioned public document. This involves incorporating 'DirAuthority' records [75] into the *torrc* configuration file. Upon initialization, the Tor relays transmit their descriptors to the private Directory Authority Servers, which subsequently record this data in a consensus file, integrating these relays into the private Tor network. Subsequently, all Tor relays download the latest consensus file, thereby finalizing the establishment of the private Tor network.

4 Experimentation and evaluation

In this section, we design experiments to verify that the Tor network simulation approach in this paper is lightweight and scalable while being able to meet the needs of de-anonymization attacks in terms of authenticity and flexibility.

4.1 Design of experiments

In this paper, the following three sets of experiments were designed:

- **Performance Experiment:** This paper analyzes the performance of the proposed method by recording the relevant parameters when simulating private Tor networks of varying sizes. In this experimental setup, the paper records parameters such as startup time and memory consumption when simulating Tor networks comprising 400, 600, 800, and 1000 Tor relays.
- **Circuit Association Experiment:** This study examines a de-anonymization attack from the user's perspective, with the objective of linking a user's IP address (the onion proxy) to the IP address of the application service they access. In contrast to existing approaches that rely on traffic correlation methods [9–14], which analyze traffic flow, this study focuses on the number of cells exchanged in the Tor circuit to correlate the relays on both ends. In Tor, cells represent the basic units of communication [78]. The experimental design is depicted in the blue box in Figure 5. The experiment started with the deployment of controlled relays equipped with a modified version of the Tor software, which enables the logging of circuit and cell data. An onion proxy was constructed using the *stem* library [79], configured to utilize two controlled relays as its Guard and Exit relays. Subsequently, the onion proxy was configured to upload a sizable file to an FTP server, thereby emulating typical user behavior within the Tor network. During this process, the circuits were monitored in real-time at the controlled Guard relay (designated "OP→Guard") and the Exit relay (labeled "Exit→S"). Following the completion of the upload, the number of cells transmitted per second for each circuit at "OP→Guard" was calculated, resulting in a vector $\mathbf{X}$ ($[X_1, X_2, \ldots, X_i, X_{i+1}, \ldots]$), where each X_i is composed of sequences like $[x_1, x_2, \ldots, x_j, x_{j+1}, \ldots]$. Similarly, the cell counts for each "Exit→S" circuit were compiled into vector $\mathbf{Y}$, represented as follows: $[Y_1, Y_2, \ldots, Y_i, Y_{i+1}, \ldots]$. To assess the correlation between vectors $\mathbf{X}$ and $\mathbf{Y}$, Equation (1) was applied. A correlation value of 1 indicates a strong association between the two variables, whereas a value of 0 indicates no correlation. The $Person(X_i, Y_j)$ represents the

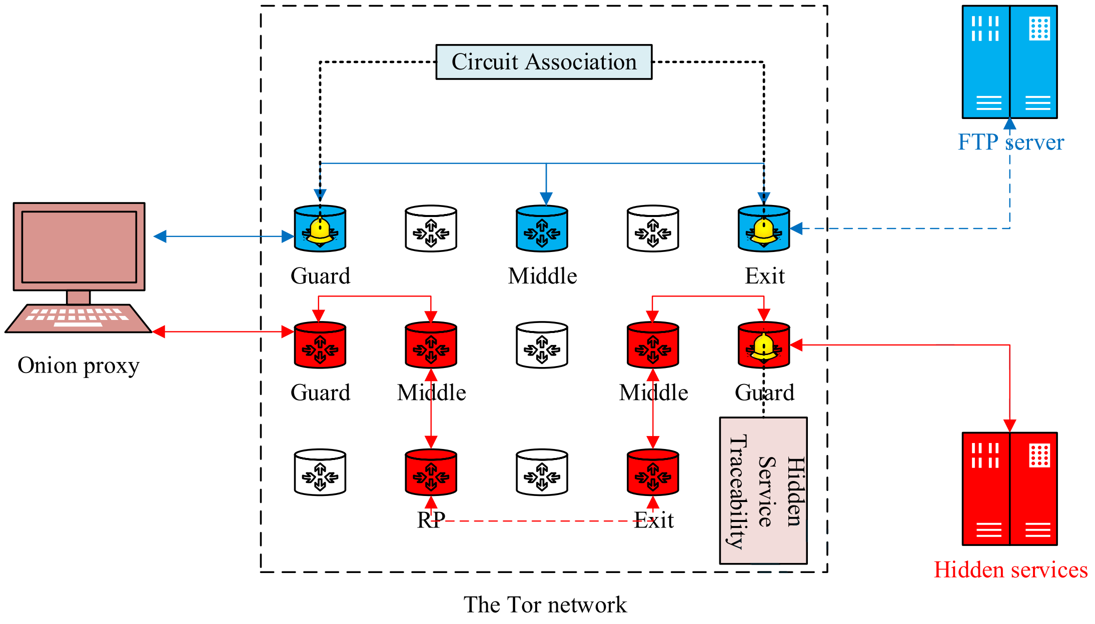

Figure 5. Circuit association and hidden service traceability scenario diagram

Pearson correlation coefficient [80] between X_i and Y_j. We consider a correlation significant when both "OP→Guard" and "Exit→S" have active circuits and the Pearson coefficient meets or exceeds 95%. This correlation indicates that the same user activity was detected at both the controlled Guard and Exit relays, which could potentially enable the tracing of the user (onion proxy) and their FTP server back to their respective IP addresses.

$$f(\mathbf{X},\mathbf{Y}) = \begin{cases} 1, & \text{if } \exists X_i \in \mathbf{X}, \exists Y_j \in \mathbf{Y}, Person(X_i, Y_j) \geq 95\% \\ 0, & \text{if } \forall X_i \in \mathbf{X}, \forall Y_j \in \mathbf{Y}, Person(X_i, Y_j) < 95\% \end{cases} \tag{1}$$

– **Hidden Service Traceability Experiment:** This experiment examines a hidden service-side de-anonymization attack that employs the WFP technique [15–17, 34, 35], which utilizes traffic fingerprinting to identify the IP address of a targeted hidden service. This method relies on the premise that different hidden services exhibit distinguishable traffic fingerprints. This experiment demonstrates the effectiveness of the proposed Tor network simulation technique in generating realistic traffic through experimental validation. The experimental setup is illustrated in the red box of Figure 5, which mirrors the WFP experiment conducted by Sirinam *et al.* [34]. Initially, we deployed ten controlled Tor relays and created ten hidden services, with each hidden service utilizing a controlled relay as its Guard relay. The content for these hidden services was derived from the home pages of the ten websites with the highest Alexa rankings. Each hidden service was accessed 512 times via an onion proxy, with the resulting traffic between the Guard and the hidden service captured in real time on the designated controlled Guard relay. We used the initial 500 accesses to train the DF model, while the final 12 accesses to test. During the training phase, the packet direction sequences were extracted from the traffic flows as features, which were then input into the DF model to construct a ten-class classification task. This was done to identify the hidden service associated with each flow. In the testing phase, the trained DF model [34] was deployed on the controlled Guard relays. This enabled the classification of user behavior when accessing hidden services. As a result, it was possible to determine which hidden services were accessed. This subsequently allowed us to trace the IP addresses of those services.

The objective of the performance experiment is to evaluate the lightweight characteristics of the Tor network simulation method, as introduced in this paper. The circuit association and hidden service traceability experiments serve to de-anonymize Tor, thereby confirming that our simulation approach meets the essential requirements for realism and flexibility. It is essential that the Tor network simulation is capable of generating authentic traffic data and allowing for real-time alterations to Tor relays.

Table 1. Table of the experimental environment

Experimental category	K8s node type	Operating system	Memory sizes	CPU core count	Hard disk storage size	Number of hosts
Performance experiment	Master	Centos 7.5	20G	8	100G	1
	Node	Centos 7.5	20G	8	100G	3
De-anonymization attacks	Master	Centos 7.9	11G	8	189G	1
	Node	Centos 7.9	11G	8	189G	1

Table 2. Table of network configuration

K8s node type	Host IP	Network segments of the simulated Tor network
Master	192.168.0.1	9.0.0.0/24
Node1	192.168.0.2	9.0.1.0/24
Node2	192.168.0.3	9.0.2.0/24
Node3	192.168.0.4	9.0.3.0/24

Moreover, we conducted performance and de-anonymization experiments with varying numbers of physical hosts to demonstrate the scalability of our method. The successful execution of these experiments demonstrates that the simulated private Tor network is operational, facilitating circuit establishment, anonymous service access, and the creation of hidden services.

4.2 Experimental results and analysis

The specifics of the experimental environment are outlined in Table 1. For the performance experiments, we utilized four hosts (Master, Node1-3), while two hosts (Master, Node1) were employed for the de-anonymization attack experiments. The scalability of the Tor network simulation method is demonstrated by varying the number of hosts. The performance experiment maintains a 2:3 ratio of Exit relays to regular relays, incorporating three Directory Authority Servers, with the total entity count ranging from 400 to 1000. In the de-anonymization attack experiments, a total of 222 entities were included, comprising 10 Directory Authority Servers, 120 regular relays, 70 Exit relays, 11 onion proxies, 10 hidden services, and 1 FTP server.

The network configuration of the experimental environment is shown in Table 2. We divide the network of the cluster hosts into 9.0.0.0/24, 9.0.1.0/24, 9.0.2.0/24, and 9.0.3.0/24 subnets in order, and based on the subnets that a relay belongs to in the private Tor network, we can determine which cluster host is running on this Tor relay. The software version of the experimental environment is Kubernetes-1.17.4, docker-18.06.3. The Tor version of the experimental environment is Directory Authority Server-0.4.6.10, Relay-0.4.6.9, Onion Proxy-0.4.4.6, and Hidden Service-0.4.4.6.

4.2.1 Performance experiment

Table 3 displays the startup times and memory usage for the simulation of Tor networks of varying sizes across four hosts, with relays distributed in a uniform manner. As the number of relays in the private Tor network increases, the overall simulation environment experiences longer startup times and higher memory usage. When the number of Tor relays reaches 1000, the startup time of the entire simulation environment reaches 4 minutes and 30 seconds, with an average memory consumption of 11.9 GB per host (47.6 GB total). This resource consumption is acceptable for simulation service providers deploying regular hosts. On average, adding 200 relays to the network increases the startup time by 1 minute and the average memory consumption per host by 2G (8G total). This indicates that the average memory usage of a Tor relay in the simulation environment is approximately 40 MB.

Compared to other simulation methods, we focus on more realistic hardware-based simulations. Because generic discrete event simulators and software-based simulations lack realism, they cannot generate realistic Tor traffic to support subsequent de-anonymization attack experiments. Hardware-based

Table 3. Table of performance experimental results

Number of simulated relays	Startup time	Average memory consumption	K8s node type	Initial memory	Memory after startup completion
400	2min 15s	4.2G	Master	18.4G	13.6G
			Node1	18.4G	14.3G
			Node2	18.4G	14.5G
			Node3	18.4G	14.3G
600	3min	6.4G	Master	18.4G	11G
			Node1	18.4G	12.4G
			Node2	18.4G	12.2G
			Node3	18.4G	12.5G
800	3min 50s	8.2G	Master	18.4G	8.9G
			Node1	18.4G	10.7G
			Node2	18.4G	10.7G
			Node3	18.4G	10.8G
1000	4min 30s	11.9G	Master	18.4G	4.4G
			Node1	18.4G	7.1G
			Node2	18.4G	7.3G
			Node3	18.4G	7.2G

simulations typically utilize a Raspberry Pi or a virtual machine as the unit for simulating Tor relays. However, due to the necessity of installing an operating system, the minimum memory requirement for these devices is typically no less than 512 MB [81, 82]. This indicates that hardware-based simulations necessitate a minimum of 100 GB of memory to simulate 200 Tor relays, far exceeding the 8 GB memory capacity of the method presented in this paper. Additionally, hardware-based simulations require additional resources, such as an operating system, CPU, and other hardware components on each Tor relay. Therefore, the Tor network simulation method proposed in this paper significantly reduces costs while providing a lightweight solution.

4.2.2 Circuit association experiment

Ten circuit association experiments were conducted, all of which resulted in successful associations, as outlined in Table 4. The results demonstrate that the Tor network simulation method effectively fulfills the requirements for realistic circuit construction and the flexible operation of Tor relays, thereby enabling normal access to the private Tor network. It is noteworthy that, due to the controlled nature of the experiments, researchers were able to fully monitor user behaviors without interference from other onion proxies, which contributed to a high success rate. In real-world scenarios, however, the uncontrollable nature of onion proxies introduces complications to data collection, making it challenging to guarantee that external traffic does not impact the results. Therefore, while user-side de-anonymization attacks encounter numerous challenges in practice, our simulation method enables researchers to reproduce intricate scenarios for comprehensive analysis.

4.2.3 Hidden service traceability experiment

In the test phase of the experiment, ten target hidden services were tracked, resulting in a total of 120 accesses. Although this results in an uneven distribution of accesses per service (not precisely 12 accesses per service), it does not affect the relevance of the experiment or the interpretation of the results. As detailed in Table 5, we successfully traced 113 instances back to the target hidden services, yielding an accuracy rate of 94.17%. This finding confirms that our Tor network simulation method meets the requirements for realistic traffic generation and flexible operation of Tor relays, as well as the accurate establishment of hidden services. Notably, we used controlled hidden services in this experiment, allowing us to assign a controlled relay as the Guard to capture traffic. However, in real-world contexts, hidden services are typically anonymous and uncontrollable, creating difficulties for data collection. Our simulation method provides a means to recreate such scenarios, offering valuable insights for researchers.

Table 4. Table of circuit association experimental results

Index	The maximum value of $Person(X_i, Y_j)$	$f(\mathbf{X}, \mathbf{Y})$
1	0.9981741464736523	1
2	0.9950434560291763	1
3	1.0	1
4	0.9973259696600676	1
5	0.9965850090502247	1
6	0.9958361479001489	1
7	0.9980901342096359	1
8	0.9952349663525573	1
9	0.9969461005607609	1
10	0.9985147913732642	1

Table 5. Table of hidden service traceability experiment results

Index	Number of access	Number of successful traceability	Individual hidden service traceability accuracy
1	12	11	0.9167
2	12	11	0.9167
3	11	11	1
4	12	12	1
5	15	13	0.8667
6	12	12	1
7	13	13	1
8	10	9	0.9
9	11	10	0.9091
10	12	11	0.9167

5 Discussion

5.1 Tor network simulation based on hybrid networking of multi-granularity relays

The Tor network simulation method in this paper meets the needs of de-anonymization attacks while being lightweight. However, in this paper, while conducting experiments on de-anonymization attacks, we discovered that numerous Tor relays remain uncontrolled by the researcher. The researcher often requires a high degree of realism for the Tor relays under his control, whereas, for the numerous uncontrolled Tor relays, only their data communication with the researcher-controlled relays is of concern.

For the Tor relays controlled by the researcher, the following three dashes of realism need to be guaranteed in the simulation: first, the relay is running a real Tor application; second, the researcher can log in to the specific relay to carry out real-time operations, such as capturing traffic data and executing the experimental scripts, *etc.*; third, the different relays are isolated from each other and their operations do not affect each other, *e.g.*, the relays have non-conflicting IP addresses, modifying the routing table of one relay will not affect the other relays, *etc.* For this type of relay, the Docker container can be used as the basic unit to simulate the Tor relay.

For the numerous uncontrolled Tor relays in the experiment, the following realism needs to be guaranteed in the simulation: firstly, their relay information needs to appear in the consensus file of the Tor network so that they can be selected to build circuits; secondly, when this type of relays communicate with the researcher-controlled relays, they can create real Tor circuits and generate communication traffic data of real Tor protocols. For this type of relay, the Docker container can be used as a secondary carrier for the relay, and a single container can be used to simulate multiple relays of this type as a way to reduce the cost of the experiment and to guarantee the necessary realism of the simulation.

In summary, this paper presents a further discussion of the Tor network simulation method based on hybrid networking of multi-granularity relays to reduce costs. By dividing the required realism of different

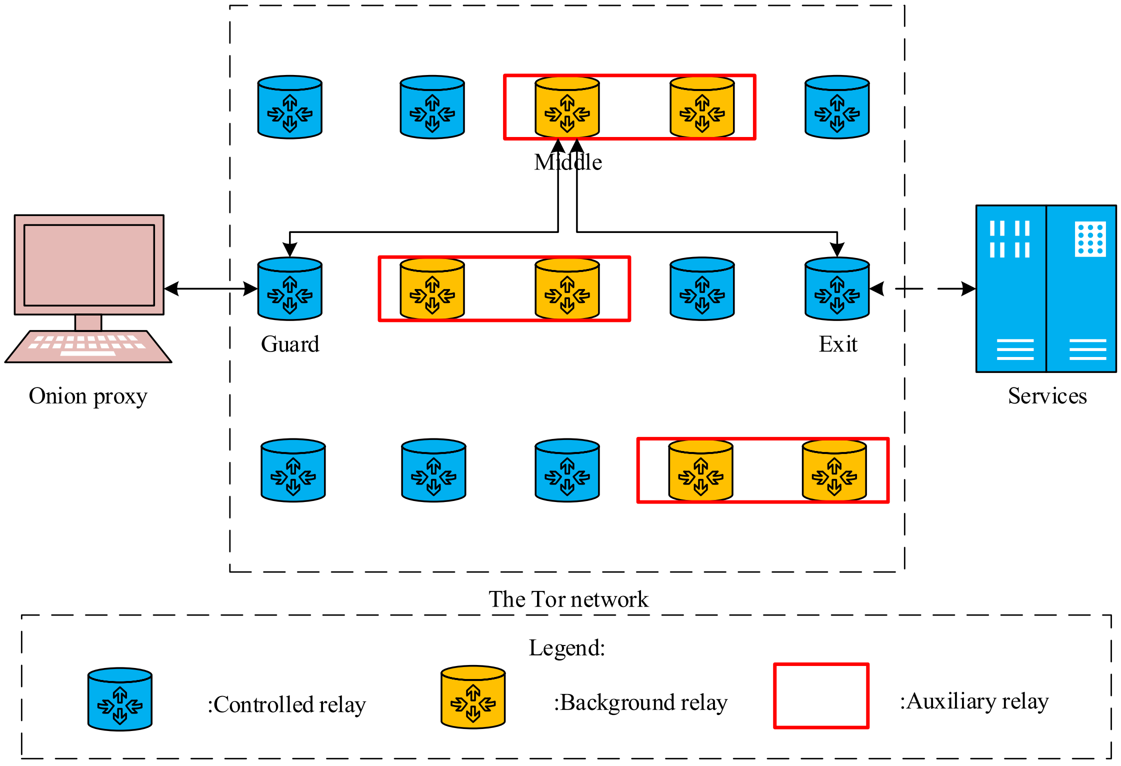

Figure 6. Architectural design of the Tor network simulation method based on hybrid networking of multi-granularity relays

relays in Tor network experiments, all relays are categorized into relays with different realism granularity. Strong realism relays guarantee the realism of the experiment, while weak realism relays reduce the cost required for the experiment.

5.1.1 Architecture design

This paper presents the architecture for designing the Tor network simulation method based on a hybrid network of multi-granularity relays, as shown in Figure 6. The Tor network includes the following three types of relays:

- **Controlled relay:** These relays are Docker containers running Tor processes and the most realistic relays in this method. The researcher can operate the relay in real-time and is able to capture real communication traffic data on the relay.
- **Background relay:** It should first be noted that background relays do not have actual Tor processes running on them. They are merely referenced in Tor's consensus file. During the simulation, these relays are incorporated into the consensus file in the format of a typical relay. It is important to highlight that these relays rely on auxiliary relays to function.
- **Auxiliary relay:** The onion proxy utilizes the consensus file to select relays for circuit construction. When a background relay is selected for circuit construction, it is replaced with an auxiliary relay to ensure the generation of real Tor protocol traffic. Auxiliary relays are constructed in the same manner as controlled relays, except that they are not under the control of the researcher.

In the Tor network simulation method based on hybrid networking of multi-granularity relays, multiple background relays correspond to 1 auxiliary relay. This correspondence ratio is indicated by the parameter p, which takes on values such as 0.1, indicating that 10 $(1/p)$ background relays correspond to 1 auxiliary relay.

5.1.2 Hybrid networking with multi-granularity relays

Multi-granularity relay hybrid networking is divided into three steps. Initially, the number of auxiliary relays required for each role is calculated based on the configuration requirements of the background relays. Subsequently, the Docker containers of the controlled and auxiliary relays are initiated. The relays then form a private Tor network by the process illustrated in Figure 4. Finally, the information on the background relays, based on the background relay's configuration requirements, is added to the consensus file generated in the previous step. Among the configuration requirements for background relays are IP address, bandwidth, relay label, ratio value (p) of background relays to auxiliary relays, and so on.

5.1.3 Hybrid establishment circuit with multi-granularity relays

The key to building circuits with a mixture of multi-granularity relays is to deal with the relationship between relays of different granularity. The controlled relay is the primary relay in the experiment and is the container in which the Tor application is running. When the relays in the circuit are all controlled relays, the Tor program's mechanism completes the establishment of the circuit. Background relays are only of interest when there is data communication with controlled relays. When the relays in the circuit are all background relays, there is no impact on the experiment. Thus, this paper simply forgoes the establishment of that type of circuit. When there are both controlled and background relays in the circuit, it is necessary to utilize auxiliary relays for the hybrid establishment of circuits.

This method requires some intervention in onion proxy routing. This method simulates routing based on the relay information of the final consensus file, and due to the presence of background relays in the consensus file, the following three scenarios are possible when routing:

- The three relays selected during the route selection process are all controlled relays. This method establishes Tor circuits directly based on the selected controlled relays and records the route selection results in the route selection log.
- The three relays selected for routing are all background relays, for which this method does not construct the actual Tor circuit. Instead, it records the routing results, as there is no controlled relay in the three-hop relay. Consequently, there is no data communication between background relays and researcher-controlled relays.
- If the three selected relays during routing have both controlled and background relays, the present method replaces one of the background relays with the corresponding auxiliary relay. Subsequently, an authentic Tor circuit from the aforementioned auxiliary relay and the selected controlled relays is constructed, and the route selection outcome is documented. To illustrate, if the first-hop, second-hop, and third-hop relays in the selected circuit are controlled relay, background relay, and controlled relay, respectively, the background relay of the second hop is replaced with an auxiliary relay, which is used as the Middle relay in the circuit to create a Tor circuit with controlled relays.

5.1.4 Experimental verification

In order to validate the effectiveness of the Tor network simulation method based on hybrid networking of multi-granularity relays, this paper involves access experiments and hybrid establishment of circuit experiments. The experimental environment is similar to the performance experiment in Table 1, with the difference that the Master node host for this experiment is 24G RAM, 16-core CPU, and 500G hard disk storage. The simulated private Tor network environment consists of three Directory Authority Servers, 600 normal Tor relays, 400 Exit relays, and 2000 background relays. The number of normal and Exit relays in the background relays is 1200 and 800, respectively. The parameter p, representing the ratio of background relays to auxiliary relays, is 0.1, indicating that 200 auxiliary relays are necessary for deployment.

In the access experiments, this paper develops ordinary web services and hidden services, both of which can be accessed successfully using the onion proxy. This result verifies that the Tor network simulation method based on hybrid networking of multi-granularity relays is functionally normal.

In the hybrid establishment circuit experiment, this paper simulates 10 000 instances of onion proxy routing, records the memory consumption of each node host during the experiment, and analyzes the routing logs. The objective is twofold: first, to verify the effectiveness of the hybrid establishment circuit

Table 6. Table on resource consumption for experiments on hybrid establishment circuits with multi-granularity relays

K8s node host	Host initial memory	Remaining host memory after private Tor network startup	Mean value of the remaining memory of the host during the hybrid establishment circuit experiments
Master	22.5G	8.4G	7G
Node1	18.4G	6.4G	5.2G
Node2	18.4G	7.2G	5.8G
Node3	18.4G	6.6G	5.3G
Node4	18.4G	6.4G	5.2G

mechanism for multi-granularity relays; second, to test the resource consumption of the hybrid networking with multi-granularity relays. The experiment employs three circuit types: Type I comprises circuits with all controlled relays; Type II, circuits with all background relays; and Type III, circuits with both controlled and background relays. In the 10 000 simulated circuit selections conducted in this experiment, the rate of type I was approximately 3.1%, the rate of type II was 27.35%, and the rate of type III was 69.53%. Table 6 illustrates the memory resource consumption of each node host during the experiment, and all circuits were successfully established. The results demonstrate that the hybrid circuit establishment mechanism based on multi-granularity relays is capable of constructing Tor circuits in a private Tor network environment in a manner that is both reasonable and realistic. Furthermore, the experimental results indicate that the Tor network simulation method based on hybrid networking of multi-granularity relays exhibits an acceptable resource cost when the number of relays reaches thousands.

5.1.5 Limitation

The Tor network simulation based on hybrid networking of multi-granularity relays, is still able to satisfy previous de-anonymization attack experiments, such as circuit association experiments and hidden service traceability experiments, because it runs the actual Tor protocols and generates the actual data, such as the actual Tor traffic. Nevertheless, this approach offers a more lightweight simulation at the expense of partial realism, which results in its inherent limitations. These limitations are reflected in the bias towards performance evaluation-type experiments, as the actual number of relays running the Tor protocol is less than the declared number. This discrepancy leads to inaccurate results when evaluating experiments such as bandwidth distribution, which is affected by the actual relay size.

5.2 Conclusion

Tor, a prominent anonymity network worldwide, has attracted more and more attention as people pay more and more attention to privacy protection. To study and analyze the Tor anonymity network, researchers have conducted numerous experiments within the Tor network. Compared to the real-world Tor network, experiments in a simulated environment do not affect the actual network's functionality and improve the overall efficiency of the experiments. In this paper, we propose a K8s-based Tor network simulation method to address the challenges of balancing scale and realism in simulating private Tor networks, as well as the lack of support for network security experiments in Tor (such as de-anonymization experiments). We experimentally prove that the method has the following characteristics: lightweight, realistic, flexible, and scalable. These experiments further verify that the method can satisfy the requirements of de-anonymization attack experiments. The simulation method proposed in this paper can simulate both Tor relays and other types of services, such as FTP servers, demonstrating a degree of generality. Furthermore, this paper presents an additional discussion of the Tor network simulation method based on hybrid networking with multi-granularity relays, which further balances realism and resource consumption.

Although the Tor network simulation method proposed in this paper addresses the issues of high simulation cost and unrealistic simulation that existed in previous research, some limitations require further development. For instance, the simulation method presented in this paper does not consider the impact of disparate geographic locations, varying bandwidths, and other variables on the latency of the Tor network. The real Tor network has thousands of relays distributed globally. The locations of different routing relays in the path chosen by the onion proxy may be geographically disparate, which will result in a significant delay in Tor communication. Furthermore, the duration of this delay is not stable. The simulated Tor network experiment platform developed in this paper currently does not provide the functionality for generating background traffic. In the real world, a wide range of network activities generates complex background traffic. How to efficiently and accurately simulate the background traffic in Tor networks will provide more realism and facilitate researchers to explore more complex environments. In future research, these factors can be incorporated into the simulation to enhance the realism of the simulated private Tor network.

Acknowledgments

Thanks to anonymous reviewers for their hard work and kind help.

Funding

This work was supported by National Key Research and Development Program of China (No. 2023YFB3106600) and National Natural Science Foundation of China (62372056).

Conflicts of interest

The authors declare no conflict of interest.

Data availability statement

No data are associated with this article.

Author contribution statement

Wentao Huang designed the framework and structure of this paper and participated in implementing the Tor network simulation method in this paper. Han Wu participated in implementing the Tor network simulation method in this paper. Zeyu Li participated in implementing the circuit association experiment in the de-anonymization attack experiment. Qingfeng Zhang participated in implementing the hidden service traceability experiment in the de-anonymization attack experiment. Xiaqing Xie and Xuebin Wang provided in-depth practical guidance on the Tor network simulation method in this paper. Jiapeng Zhao improved the readability of this paper by grammatical modification and polishing. Jinqiao Shi provided important guidance on the implementation of the method and paper writing.

References

[1] Dingledine R, Mathewson N, Syverson PF, et al. Tor: The second-generation onion router. In: USENIX Security Symposium, vol. 4, 2004, 303–320.
[2] Inc., T.T.P. Tor Metrics Portal, https://metrics.torproject.org/, last accessed 4 Aug. 2024.
[3] Upadhyaya R and Jain A. Cyber ethics and cyber crime: A deep dwelved study into legality, ransomware, underground web and bitcoin wallet. In: 2016 International Conference on Computing, Communication and Automation (ICCCA), 2016, 143–148.
[4] Ghafir I, Svoboda J and Prenosil V. Tor-based malware and tor connection detection. In: International Conference on Frontiers of Communications, Networks and Applications (ICFCNA 2014 – Malaysia), 2014, 1–6.
[5] Salas Conde JJ, Martín Ortíz M and Carneiro Díaz VM. Methodology for identification and classifying of cybercrime on tor network through the use of cryptocurrencies based on web textual contents. Computación y Sistemas 2022; **26**: 347–356.
[6] Nurmi J. Understanding the usage of anonymous onion services: Empirical experiments to study criminal activities in the tor network, 2019.
[7] Jardine E. The dark web dilemma: Tor, anonymity and online policing. Global Commission on Internet Governance Paper Series, 2015.
[8] Saleem J, Islam R and Kabir MA. The anonymity of the dark web: A survey. IEEE Access 2022; **10**: 33628–33660.
[9] Galteland H and Gjøsteen K. Adversaries monitoring tor traffic crossing their jurisdictional border and reconstructing tor circuits. arXiv preprint `arXiv:1808.09237`, 2018.
[10] Nasr M, Bahramali A and Houmansadr A. Deepcorr: Strong flow correlation attacks on tor using deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, 1962–1976.
[11] Li J, Gu C, Zhang X, et al. Attcorr: A novel deep learning model for flow correlation attacks on tor. In: 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE), IEEE, 2021, 427–430.
[12] Oh SE, Yang T, Mathews N, et al. Deepcoffea: Improved flow correlation attacks on tor via metric learning and amplification. In: 2022 IEEE Symposium on Security and Privacy (SP), IEEE, 2022, 1915–1932.

[13] Guan Z, Liu C, Xiong G, et al. Flowtracker: Improved flow correlation attacks with denoising and contrastive learning. Comput Secur 2023; **125**: 103018.
[14] Lopes D, Dong JD, Medeiros P, et al. Flow correlation attacks on tor onion service sessions with sliding subset sum, 2024.
[15] Wang M, Li Y, Wang X, et al. 2ch-TCN: A website fingerprinting attack over tor using 2-channel temporal convolutional networks. In: 2020 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2020, 1–7.
[16] Bang J, Jeong J and Lee J. Fedfingerprinting: A federated learning approach to website fingerprinting attacks in tor networks. IEEE Access 2023.
[17] Song C, Fan Z, Wang H, et al. Seamless website fingerprinting in multiple environments. arXiv preprint arXiv:2407.19365, 2024.
[18] Chen M, Wang X, Shi J, et al. Napping guard: Deanonymizing tor hidden service in a stealthy way. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE, 2020, 699–706.
[19] Li T, Liu K, Feng W, et al. Heterotic: A robust network flow watermarking based on heterogeneous time channels. Comput Netw 2022; **219**: 109424.
[20] Zhang Q, Teng Z, Wang X, et al. Hsdirsniper: A new attack exploiting vulnerabilities in tor's hidden service directories. In: Proceedings of the ACM on Web Conference 2024, 2024, 1812–1823.
[21] Shirazi F, Goehring M and Diaz C. Tor experimentation tools. In: 2015 IEEE Security and Privacy Workshops, IEEE, 2015, 206–213.
[22] Wang Q and Cao W. A tor anonymity attack experiment platform driven by Raspberry Pi. In: 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan), IEEE 2020, 569–574.
[23] Anderson C. Docker [software engineering]. IEEE Softw 2015; **32**: 102–c3.
[24] Google: K8s: Production-grade container orchestration, https://kubernetes.io/, last accessed 4 Aug. 2023.
[25] Project T. Tor Browser, https://www.torproject.org/download/, last accessed 10 June 2023.
[26] Mathewson, N. Tor Rendezvous Specification, https://gitlab.torproject.org/tpo/core/torspec/-/blob/HEAD/rend-spec-v3.txt, last accessed 12 June 2023.
[27] William P, Jawale M, Pawar A, et al. Systematic approach for detection and assessment of dark web threat evolution. In: Using Computational Intelligence for the Dark Web and Illicit Behavior Detection, IGI Global, 2022, 230–256.
[28] Karunanayake I, Ahmed N, Malaney R, et al. De-anonymisation attacks on tor: A survey. IEEE Commun Surveys Tutor 2021; **23**: 2324–2350.
[29] Guan Z, Xiong G, Li Z, et al. Restor: A pre-processing model for removing the noise pattern in flow correlation. In: 2020 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2020, 1–6.
[30] Tian J, Gou G, Guan Y, et al. Universal perturbation for flow correlation attack on tor. In: 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC), IEEE, 2021, 1–9.
[31] Hafeez MA, Ali Y, Han KH, et al. GPU-accelerated deep learning-based correlation attack on tor networks. IEEE Access 2023.
[32] Guan Z, Liu C, Gou G, et al. A blind flow fingerprinting and correlation method against disturbed anonymous traffic based on pattern reconstruction. Comput Netw 2024; **254**: 110831.
[33] Schroff F, Kalenichenko D and Philbin J. Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, 815–823.
[34] Sirinam P, Imani M, Juarez M, et al. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, 1928–1943.
[35] Cherubin G, Jansen R and Troncoso C. Online website fingerprinting: Evaluating website fingerprinting attacks on tor in the real world. In: 31st USENIX Security Symposium (USENIX Security 22), 2022, 753–770.
[36] Kwon A, AlSabah M, Lazar D, et al. Circuit fingerprinting attacks: Passive deanonymization of tor hidden services. In: 24th USENIX Security Symposium (USENIX Security 15), 2015, 287–302.
[37] Hayes J and Danezis G. k-fingerprinting: A robust scalable website fingerprinting technique. In: 25th USENIX Security Symposium (USENIX Security 16), 2016, 1187–1203.
[38] Panchenko A, Mitseva A, Henze M, et al. Analysis of fingerprinting techniques for tor hidden services. In: Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, 2017, 165–175.
[39] Henri S, Garcia-Aviles G, Serrano P, et al. Protecting against website fingerprinting with multihoming. Proc Priv Enhancing Technol 2020.
[40] Xu Y, Wang L, Li J, et al. Zoomer: A website fingerprinting attack against tor hidden services. In: International Conference on Information and Communications Security, Springer, 2023, 370–382.

[41] Wang M, Chen M, Li Z, et al. Deanonymize tor hidden services using remote website fingerprinting. In: 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE, 2023, 998–1005.

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

[43] Ling Z, Luo J, Xu D, et al. Novel and practical sdn-based traceback technique for malicious traffic over anonymous networks. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, 1180–1188.

[44] Chen M, Wang X, Liu T, et al. Signalcookie: Discovering guard relays of hidden services in parallel. In: 2019 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2019, 1–7.

[45] Iacovazzi A, Frassinelli D and Elovici Y. The {DUSTER} attack: Tor onion service attribution based on flow watermarking with track hiding. In: 22nd International Symposium on Research in Attacks, Intrusions and Defenses, (RAID 2019), 2019, 213–225.

[46] Zhang Q, Chen M, Wang X, et al. Knock-knock: De-anonymise hidden services by exploiting service answer vulnerability. In: 2024 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2024, 1–7.

[47] Project T. How Often Does Tor Change its Paths? https://support.torproject.org/about/change-paths/, last accessed 15 June 2023.

[48] O'Gorman G and Blott S. Simulating low-latency anonymous networks. In: Proceedings of the 2009 Spring Simulation Multiconference, 2009, 1–8.

[49] Doswell S, Aslam N, Kendall D, et al. The novel use of bridge relays to provide persistent tor connections for mobile devices. In: 2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), IEEE, 2013, 3371–3375.

[50] Musa A and Awan I. Functional and performance analysis of discrete event network simulation tools. Simul Model Practice Theory 2022; **116**: 102470.

[51] Jansen R and Hopper NJ. Shadow: Running tor in a box for accurate and efficient experimentation, 2011.

[52] Bauer K, Sherr M and Grunwald D. {ExperimenTor}: A testbed for safe and realistic tor experimentation. In: 4th Workshop on Cyber Security Experimentation and Test (CSET 11) 2011.

[53] Vahdat A, Yocum K, Walsh K, et al. Scalability and accuracy in a large-scale network emulator. ACM SIGOPS Operating Syst Rev 2002; **36**: 271–284.

[54] Wacek C, Tan H, Bauer KS, et al. An empirical evaluation of relay selection in tor. In: NDSS, 2013.

[55] AlSabah M, Bauer K, Elahi T, et al. The path less travelled: Overcoming tor's bottlenecks with traffic splitting. In: Privacy Enhancing Technologies: 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10–12, 2013. Proceedings 13, Springer, 2013, 143–163.

[56] Elahi T, Bauer K, AlSabah M, et al. Changing of the guards: A framework for understanding and improving entry guard selection in tor. In: Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society, 2012, 43–54.

[57] Johnson A, Wacek C, Jansen R, et al. Users get routed: Traffic correlation on tor by realistic adversaries. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, 2013, 337–348.

[58] Singh S. Large-scale emulation of anonymous communication networks. Master's Thesis, University of Waterloo, 2014.

[59] Jansen R, Geddes J, Wacek C, et al. Never been {KIST}:{Tor's} congestion management blossoms with {Kernel-Informed} socket transport. In: 23rd USENIX Security Symposium (USENIX Security 14), 2014, 127–142.

[60] Jansen R, Tracey J and Goldberg I. Once is never enough: Foundations for sound statistical inference in tor network experimentation. In: 30th USENIX Security Symposium (USENIX Security 21), 2021, 3415–3432.

[61] Jansen R, Newsome J and Wails R. Co-opting linux processes for {High-Performance} network simulation. In: 2022 USENIX Annual Technical Conference (USENIX ATC 22), 2022, 327–350.

[62] Cheng CY. High-fidelity simulation of load balancing methods in tor. Ph.D. Thesis, University of Illinois at Urbana-Champaign, 2020.

[63] Chun B, Culler D, Roscoe T, et al. Planetlab: An overlay testbed for broad-coverage services. ACM SIGCOMM Comput Commun Rev 2003; **33**: 3–12.

[64] Bauer K, McCoy D, Grunwald D, et al. Low-resource routing attacks against tor. In: Proceedings of the 2007 ACM workshop on Privacy in Electronic Society, 2007, 11–20.

[65] Akhoondi M, Yu C and Madhyastha HV. Lastor: A low-latency as-aware tor client. In: 2012 IEEE Symposium on Security and Privacy, 2012, IEEE, 476–490.

[66] Benzel T. The science of cyber security experimentation: The deter project. In: Proceedings of the 27th Annual Computer Security Applications Conference, 2011, 137–148.

[67] Mirkovic J, Benzel TV, Faber T, et al. The deter project: Advancing the science of cyber security experimentation and test. In: 2010 IEEE International Conference on Technologies for Homeland Security (HST), IEEE, 2010, 1–7.

[68] Chakravarty S, Stavrou A and Keromytis AD. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In: European Symposium on Research in Computer Security, Springer, 2010, 249–267.

[69] White B, Lepreau J, Stoller L, et al. An integrated experimental environment for distributed systems and networks. ACM SIGOPS Operating Syst Rev 2002; **36**: 255–270.

[70] Das A and Borisov N. Securing anonymous communication channels under the selective dos attack. In: Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1–5, 2013, Revised Selected Papers 17, Springer, 2013, 362–370.

[71] Wang P, Liu H, Wang B, et al. Simulation of dark network scene based on the big data environment. In: Proceedings of the International Conference on Information Technology and Electrical Engineering, 2018, 1–6.

[72] Zheng X, Yan H, Wang R, et al. Lunar: A practical anonymous network simulation platform. Secur Commun Netw 2022; **2022**: 5832124.

[73] Hat R: Understanding Virtualization, https://www.redhat.com/en/topics/virtualization, last accessed 15 July 2023.

[74] dockercon: Dockerfile Reference, https://docs.docker.com/engine/reference/builder/, last accessed 15 Sep. 2023.

[75] Project T: Tor-stable Manual, https://2019.www.torproject.org/docs/tor-manual.html.en, last accessed 17 Sep. 2023.

[76] feiskyer: kubeadm, https://kubernetes.feisky.xyz/setup/cluster/kubeadm, last accessed 18 Sep. 2023.

[77] rbrtbnfgl: flannel, https://github.com/flannel-io/flannel, last accessed 19 Sep. 2023.

[78] Mathewson N. Tor's Protocol Specifications, https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt, last accessed 12 June 2023.

[79] Docs S. Api, https://stem.torproject.org/api.html, last accessed 18 Sep. 2023.

[80] Sedgwick P. Pearson's Correlation Coefficient. BMJ, 2012.

[81] Debian: Installing Debian GNU/Linux from a Unix/Linux System, https://www.debian.org/releases/stable/amd64/apds03.en.html, last accessed 10 June 2024.

[82] Pi R. Raspberry Pi, https://www.raspberrypi.org/products/, last accessed 10 June 2024.

Wentao Huang received his B.S. degree from Beijing Information Science and Technology University in 2021 and is currently pursuing his Ph.D. degree at Beijing University of Posts and Telecommunications. His research interests include anonymous communication and encrypted traffic analysis.

Han Wu graduated from the School of Cyberspace Security, Beijing University of Posts and Telecommunications with a master's degree in 2023, and his main research interests are anonymous communications.

Zeyu Li received his master's degree from Chongqing Posts and Telecommunications University in 2020, and is currently pursuing his Ph.D. degree at Beijing Posts and Telecommunications University. His current research interests are in encrypted traffic analysis.

Xiaqing Xie is currently working in Beijing University of Posts and Telecommunications, mainly engaged in data security and network security related research work.

Qingfeng Zhang received his PhD from the Institute of Information Engineering of the University of Chinese Academy of Sciences in 2024, and has been engaged in anonymous communication related research for a long time.

Xuebin Wang is a senior engineer at the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include anonymous communication, blockchain security, and decentralized social networks.

Jiapeng Zhao received his Ph.D. degree in information security from University of Chinese Academy of Sciences, Beijing, China, in 2022. He is currently a postdoctor of information security with Beijing University of Posts and Telecommunications (BUPT). His current research interests include dark web, network traffic analysis, cyberspace security and nature language processing.

Jinqiao Shi is a professor and doctoral supervisor at the School of Cyberspace Security, Beijing University of Posts and Telecommunications. His main research interests include distributed network system, anonymous communication and privacy protection, blockchain network, intelligent information processing, intelligent analysis of big data, network measurement technology, *etc.*