EqualNet: A Secure and Practical Defense for Long-term Network Topology Obfuscation

Jinwoo Kim*, Eduard Marin†, Mauro Conti‡, and Seungwon Shin*
*School of Electrical Engineering, KAIST
†Telefonica Research
‡Department of Mathematics, University of Padua
{jinwoo.kim, claude}@kaist.ac.kr eduard.marinfabregas@telefonica.com conti@math.unipd.it

***Abstract*—Path tracing tools, such as traceroute, are simple yet fundamental network debugging tools for network operators to detect and fix network failures. Unfortunately, adversaries can also use such tools to retrieve previously unknown network topology information which is key to realizing sophisticated Denial-of-Service attacks, such as Link Flooding Attacks (LFAs), more efficiently. Over the last few years, several network obfuscation defenses have been proposed to proactively mitigate LFAs by exposing virtual (fake) topologies that conceal potential bottleneck network links from adversaries. However, to date there has been no comprehensive and systematic analysis of the level of security and utility their virtual topologies offer. A critical analysis is thus a necessary step towards better understanding their limitations and building stronger and more practical defenses against LFAs.**

In this paper, we first conduct a security analysis of the three state-of-the-art network obfuscation defenses. Our analysis reveals four important, common limitations that can significantly decrease the security and utility of their virtual topologies. Motivated by our findings, we present EqualNet, a secure and practical proactive defense for long-term network topology obfuscation that alleviates LFAs within a network domain. EqualNet aims to equalize tracing flow distributions over nodes and links so that adversaries are unable to distinguish which of them are the most important ones, thus significantly increasing the cost of performing LFAs. Meanwhile, EqualNet preserves subnet information, helping network operators who use path tracing tools to debug their networks. To demonstrate its feasibility, we implement a full prototype of it using Software-Defined Networking (SDN) and perform extensive evaluations both in software and hardware. Our results show that EqualNet is effective at equalizing the tracing flow distributions of small, medium and large networks even when only a small number of routers within the network support SDN. Finally, we analyze the security of EqualNet against a wide variety of attacks.

I. Introduction

It is widely acknowledged that Distributed Denial of Service (DDoS) attacks constitute one of the Internet's major threats today. In recent years, DDoS attacks have been aggravated by the rise in the number of insecure Internet-of-Things (IoT) devices connected to the Internet. In the literature, there exist two main types of DDoS attacks: (i) those that target end-hosts and servers (i.e., *volume-based attacks*) and (ii) those that target the network infrastructure, also known as *Link Flooding Attacks* (LFAs). Volume-based attacks are simple yet effective attacks whose goal is to overload a server by sending a large volume of traffic to it. One prominent example of such an attack is the Mirai botnet [14], which affected many popular websites such as Twitter or Netflix [1], [3], [2]. Fortunately, these attacks can now be prevented to a large extent through the use of Content Delivery Network (CDN) infrastructures [13]. Instead, LFAs aim to disrupt the network connectivity of as many users as possible by congesting network links [42], [31]. The goal of adversaries is to inject a large amount of (separate) flows such that they all simultaneously traverse a set of *core* network links to overload them. For this purpose, adversaries can use *low-volume, separate flows* that are *indistinguishable* from normal traffic, making it difficult for network operators to develop defenses to protect against LFAs. Note that adversaries do not require knowledge of any information about the capacity or the load of links and routers to conduct LFAs.

According to Meier et al. [37], executing a LFA against an *arbitrary link* without knowing the network topology requires five times more flows than when the adversary knows the network topology. Similarly, the number of flows needed to perform a LFA against a *target link* is orders of magnitude higher when the topology is not known. Indeed, having some knowledge of the network topology is an important prerequisite to execute *effective*, *efficient* and *stealthy* LFAs. This is important for adversaries, as their goal is always to conduct attacks that cause significant damage while minimizing the cost of their attacks (i.e., the number of flows they have to create) and the chances of being detected as much as possible.

At first glance, one could argue that keeping the network topology confidential could be an effective mechanism for network operators to increase the cost to perform successful LFAs. This defense would align well with today's Internet Service Providers (ISPs) since they usually regard their network topologies as confidential [25], [40]. However, researchers have shown that existing *path-tracing tools*, such as traceroute, can be used to reveal previously unknown ISPs' network topologies including their forwarding behavior and *tracing flow distributions*, i.e., the number of traceroute flows received by each router's interface [30]. Hence, one can expect that adversaries will apply these techniques to carry out more efficient and effective LFAs [31].

Over the last few years, researchers have proposed several *reactive* [36], [29], [47], [45], [20] and *proactive* countermeasures [37], [46], [43], [15] against LFAs. As reactive defense

Network and Distributed Systems Security (NDSS) Symposium 2022
24-28 April 2022, San Diego, CA, USA
ISBN 1-891562-74-6
https://dx.doi.org/10.14722/ndss.2022.23154
www.ndss-symposium.org

methods can only detect such attacks, we focus solely on proactive countermeasures (i.e., network topology obfuscation solutions) that mitigate such attacks by exposing a virtual (false) network topology that conceals potential bottleneck links and nodes while in some cases also attempting to maintain the utility of the information provided by path tracing tools. Note that proactive countermeasures do *not* prevent LFAs, but rather significantly increase the cost to realize them, thus also reducing the incentives to execute such attacks.

Despite their efforts, we discovered four important weaknesses in the three state-of-the-art network obfuscation solutions which can significantly lower the *security* and *utility* of their virtual topologies. Motivated by our findings, we propose EqualNet, a novel proactive network topology obfuscation defense that alleviates LFAs *within* an Autonomous System (AS)[1]. The fundamental goal of EqualNet is to prevent adversaries who use path tracing tools from gaining any advantage over blind (inefficient) LFAs executed without such tools. We make the crucial observation that LFAs benefit from the fact that some nodes and links in the network appear more than others in the tracing responses obtained by adversaries. EqualNet aims to *equalize the path tracing flow distribution* over network nodes and links, such that adversaries are convinced that all nodes and links are equally *"popular"* (i.e., important) in the network. Meanwhile, EqualNet maintains the utility of the information collected by path tracing tools.

To demonstrate its feasibility, we implement EqualNet using Software-Defined Networking (SDN) and conduct extensive evaluations both using software simulations and a hardware testbed. Our evaluation shows that EqualNet is effective at hiding the popularity of nodes and links in small, medium and large networks at the cost of creating a reasonable number of virtual nodes and while preserving the utility of path tracing information. Overall, EqualNet allows for a significant reduction of the topology leakage and hence increases the cost of performing successful LFAs. Through experiments, we demonstrate that EqualNet performs well even when only a small number of routers support SDN. Finally, we show that EqualNet can resist a variety of security attacks.

Our contributions

- We analyze the three state-of-the-art proactive network obfuscation defenses, namely NetHide [37], Trassare et al.'s solution [43] and LinkBait [46]. This results in the identification of four common weaknesses which can be used to significantly lower the security and utility of the virtual topologies they expose (Section III).
- We propose EqualNet, a novel proactive network obfuscation solution that can mitigate LFA attacks within an AS. EqualNet conceals popular nodes and links by generating virtual topologies with equalized path tracing flow distributions which preserve the utility of the path tracing information (Section IV and V).
- We implement a full prototype of EqualNet using Software-Defined Networking and carry out experiments both in software and hardware to evaluate its feasibility (Section VI). Finally, we provide a detailed security analysis of EqualNet (Section VII).

[1]The protection of links between ASes is out of scope for this paper.

II. Path tracing tools

One of the most widely used path-tracing tools today is *traceroute* [12], which is used to track the route IP packets take on their way from a source to a destination host. With traceroute, the source sends the destination host a series of tracing packets whose Time-To-Live (TTL) field is increased by 1 each time, starting with TTL=1. Each router along the path decreases the TTL by 1, checks whether TTL is equal to zero and if so sends an *ICMP_time_exceeded* packet containing the router's ingress IP address and Round-Trip Time (RTT) information to the source host. This process is repeated until the source host receives an *ICMP_port_unreachable*, caused by the packet reaching the host or the maximum number of hops allowed.

Path tracing tools like traceroute are still today widely used by network operators to identify and locate failures in their neighbor ASes. Upon detecting a failure, they typically notify network operators of the neighbor AS and provide them with tracing information so that they can quickly resolve the problem [6], [41]. Unfortunately, experience has shown that adversaries can also use path tracing tools to obtain previously unknown network topologies, and even more, identify core nodes and links in the network, i.e., interesting targets to launch LFAs [31], [42], [30]

III. Problem statement and motivation

In this section, we start by introducing the three state-of-the-art network obfuscation solutions we analyze in this paper. Then, through a running example, we show the information adversaries can obtain from the tracing responses when network operators deploy each of the state-of-the-art network obfuscation solutions separately (see Figure 1). This is followed by a rigorous analysis on the security and utility the virtual topologies generated by existing solutions offer.

In the rest of this paper, we distinguish between two types of network topology: (i) the *physical topology* and (ii) the *logical topology*. The former consists of the routers and the (physical) links between them, whose configuration is only known to network operators. Instead, the latter comprises a set of logical nodes (i.e., the routers' ingress interfaces) and the interconnections between them, i.e., the logical links. This is the network topology view adversaries can obtain through path tracing tools.

Existing network obfuscation solutions. The goal of proactive network obfuscation solutions is to conceal potential bottleneck links from adversaries by exposing *secure* virtual network topologies while (in some cases also) maintaining the *utility* of the path tracing information[2]. NetHide [37] is a representative research work which generates secure virtual topologies that conceal potential bottleneck links from adversaries while retaining most of the utility of the path tracing information. To achieve this, NetHide first selects a (small) set of secure[3] virtual topologies and among them chooses the

[2]Note that NetHide is the only solution that considers utility when selecting its virtual topologies; the other works focus solely on the security aspect.

[3]For the authors of NetHide, a virtual topology is secure if it guarantees that the number of active flows that traverse potential bottleneck links are below their capacity (so that those links are not congested).

most useful[4] one. Similarly, Trassare et al.'s [43] proposed a topology obfuscation solution to identify and protect the node whose failure can cause the greatest impact within the network. To find such a node, the authors rely on the usage of the betweenness centrality metric. In particular, their solution generates all possible virtual topologies – each of them with only a single virtual link between each pair of nodes – and among the resulting virtual topologies it chooses the one that most minimizes the betweenness centrality of the most important node. Another prominent solution is LinkBait [46], which protects potential bottleneck links by rerouting some tracing flows either to nearby links (i.e., branch links) or non-bottleneck links (i.e., bait links). The authors claim that bait links can be branch links too. By rerouting tracing flows, LinkBait can reduce the number of times bottleneck links appear in tracing responses, tricking the adversary into thinking that the bottleneck link is somewhere else in the network.

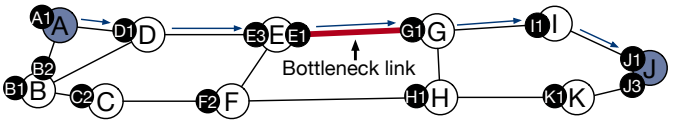
Running example. Imagine an adversary who wants to discover interesting targets to conduct LFAs leveraging the information obtained via path tracing tools (see Figure 1a). The adversary controls many hosts spread across the network and uses such hosts to send a large number of tracing packets to the network. For the sake of simplicity, let us consider only one of such tracing flows; e.g., one that originates from a host connected to node A and terminates in a host connected to node J, leading to the following tracing response: A1 $\rightarrow$ D1 $\rightarrow$ E3 $\rightarrow$ G1 $\rightarrow$ I1 $\rightarrow$ J1. Note that the numbers after the nodes denote the routers' ingress interface, e.g., A1 corresponds to the IP address of interface 1 in node A. Now assume the adversary finds that link E1-G1 (i.e., the logical link E3-G1) is frequently observed in the tracing responses she obtains, and thus decides to execute a LFA targeting this link. Next, we detail how each of the existing topology obfuscation solutions could conceal link E1-G1 from adversaries.

NetHide. Figure 1b shows a possible virtual topology that NetHide could deploy to conceal link E1-G1[5]. The core idea behind NetHide's strategy is to allow nodes E and G to appear in tracing responses as long as they do not do this consecutively. To avoid the latter, NetHide modifies the TTL field of all tracing packets that traverse node E and expire at node G (and vice versa) in order to skip one of these nodes. Hence, the adversary would see either A1 $\rightarrow$ D1 $\rightarrow$ G1 $\rightarrow$ I1 $\rightarrow$ J1 or A1 $\rightarrow$ D1 $\rightarrow$ E3 $\rightarrow$ I1 $\rightarrow$ J1 in her traceroute response.

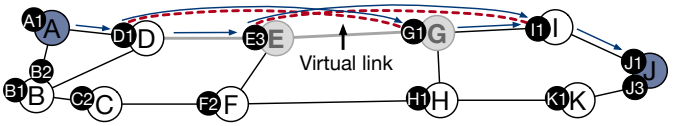
Trassare et al's solution. Unlike NetHide where all routers (except for node E or G) reply to expired tracing packets, in Trassare et al.'s solution, tracing packets are always answered by the ingress router (i.e., node A). Figure 1c shows a possible virtual topology that Trassare et al's solution could generate to conceal link E1-G1 containing only a single virtual link. If this defense is deployed, the adversary would obtain the traceroute response A1 $\rightarrow$ D1 $\rightarrow$ G1 $\rightarrow$ I1 $\rightarrow$ J1.

[4]The usefulness of a virtual topology is a measure that reflects its *accuracy* and *utility*. The accuracy reflects how similar the network paths are in the real and virtual topologies, respectively, while the utility is an indicator of how physical events (e.g., link failures and congestion) are observed in the virtual topology (and vice versa).

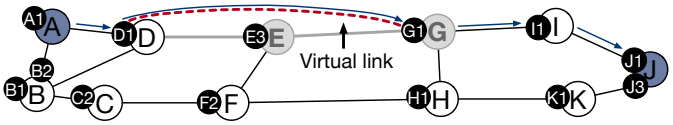
[5]Note that this virtual topology represents the best case for NetHide since only one real node is skipped. It is worth noting that in practice NetHide may require to skip more than one real node in order to generate a secure virtual topology.



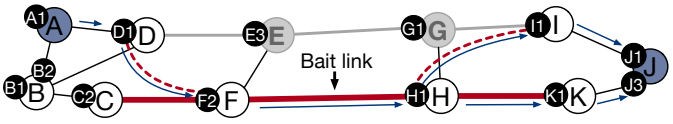
(a) Abilene network topology (used as a reference in our example). The black circles denote router interfaces and the red link denotes a bottleneck link.



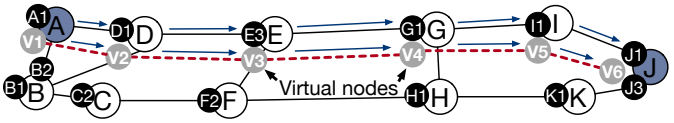
(b) Possible **NetHide's** virtual topology. Tracing flows are skipped on node E or node G, which results in a virtual link D1-G1 or E3-I1 (red dashed lines). The grey nodes (E3 and G1) and the grey links cannot be seen by adversaries.



(c) Possible **Trassare et al's** virtual topology. Tracing flows are all handled by the ingress router (i.e., node A) and are skipped on node E, which results in a virtual link D1-G1 (the red dashed line). The grey node (E3) and the grey links cannot be seen by adversaries.



(d) Possible **LinkBait's** virtual topology. LinkBait reroutes some tracing flows to one of the bait links, i.e., C-F, F-H or H-K (red lines). Dashed red lines denote virtual links.



(e) Possible **EqualNet's** virtual topology. The virtual node V4 and its neighbor virtual nodes (V1, V2, V3, V5 and V6) are added to form virtual links V1-V2, V2-V3, V3-V4, V4-V5 and V5-V6. Tracing flows are equalized over real and virtual nodes.

Fig. 1. Comparison between virtual topology generation procedures of previous work and EqualNet.

LinkBait. Figure 1d depicts a possible virtual topology that LinkBait could expose to hide link E1-G1. Suppose the bait link comprises three links: C-F, F-H and H-K which are selected on the basis of being the closest links to link E-G with sufficient available bandwidth. LinkBait picks the most suitable link (based on its current latency) within the bait link (e.g., link C-F). Eventually, if link C-F starts to receive a large amount of traffic, LinkBait automatically selects another link within the group (e.g., F-H). Overall, if LinkBait is used, the adversary would obtain a traceroute response containing the sequence A1 $\rightarrow$ D1 $\rightarrow$ F2 $\rightarrow$ H1 $\rightarrow$ I1 $\rightarrow$ J1.

Limitations of existing solutions. After a thorough systematic analysis of the three state-of-the-art network obfuscation defenses, we identified four fundamental weaknesses in all of them. Note that fixing these weaknesses would require major changes in existing defenses or even fully redesigning them. Next, we describe each of these limitations in more detail.

1) Adversaries can infer the popularity of nodes and links. Previous work focused solely on identifying and concealing individual links that could become a bottleneck from adversaries. However, they did not look at this problem from the perspective of an adversary who collects a large amount of tracing responses and makes inferences about the popularity of logical *nodes* (i.e., router interfaces) and links. This led to the design of solutions that can hide certain bottleneck links from adversaries, but are unable to conceal the number of appearances of logical nodes in tracing responses. The latter is problematic because, even if virtual topologies are deployed, adversaries can still execute efficient LFAs by targeting those nodes that appear more frequently in their tracing responses.

EqualNet. Unlike previous solutions, EqualNet tackles a much more fundamental and challenging problem, that is, preventing *imbalanced tracing flow distributions* in the adversary's view of the network topology. In particular, EqualNet analyzes the view of the network topology from the adversary perspective in order to determine and conceal the popularity of logical links and nodes so that adversaries are unable to distinguish which of them are the most popular ones.

2) Virtual topologies are too similar to real topologies. Existing solutions aim to generate secure virtual topologies on top of real nodes. In particular, to conceal a bottleneck link, these solutions can either (i) create virtual links, (ii) skip real links and nodes or (iii) use a combination of the previous methods. However, none of them considered the addition of virtual nodes in order to increase the complexity of the virtual network topology observed by the adversary. Without adding virtual nodes, regardless of the network size, the number of potential virtual topologies that can be generated is often very limited. This results in virtual topologies that are too similar to the real topologies and thus easier to reverse. Because of this, the topology leakage in the virtual topology is also likely to be very similar to the one in the real topology.

Additionally, adversaries can leverage the fact that the process of generating virtual topologies in existing solutions is (to a large extent) deterministic, to infer information about the real topology. For example, the problem of NetHide is that it can only choose a virtual topology at random from the (limited) set of network topologies that are both secure and most useful. Depending on the number of virtual topologies that fulfill both conditions, the randomness in the process of choosing a virtual topology can be significantly reduced. This also applies to Trassare et al's solution which always exposes the virtual topology (with a single virtual link) that most minimizes the betweenness centrality of the bottleneck node. Equally, the procedure by which LinkBait finds bait links can be inferred by the adversary since LinkBait does not provide many options to choose bait links. For example, in Figure 1d, bait links can only be chosen from a disjointed path of the original path (e.g., $B1 \rightarrow C2 \rightarrow F2 \rightarrow H1 \rightarrow K1 \rightarrow J3$) in order to avoid selecting the link where the bottleneck is. In all these cases, adversaries can leverage the deterministic nature of the virtual topology generation to make guesses about the real topology (including where the bottleneck link can be).

EqualNet. EqualNet not only creates virtual links and skips real ones (like previous work), but also extends the network topology by creating virtual nodes (indistinguishable from real nodes) that are connected to each other, forming fully disjoint paths from those that connect real logical nodes. The addition of virtual nodes is fundamental to increase the number of virtual topologies that can be exposed (which increases the difficulty of attacks). Additionally, EqualNet relies on a non-deterministic algorithm for adding virtual nodes which selects IP addresses at random (within the subnet of the real node's IP address).

3) Virtual topologies are not secure long-term. Previous solutions focus solely on generating a *single instance* of a secure virtual topology. Moreover, none of the existing solutions considers the *topology leakage* exposed to adversaries who can observe slightly modified virtual topologies deployed at different times. This is important because the presence or absence of a given node in two consecutive virtual topologies can disclose key insights about potential bottlenecks within the network which can ultimately facilitate LFAs.

EqualNet. EqualNet continuously monitors how much topology information is leaked, and dynamically re-obfuscates the network whenever the topology leakage exceeds the obfuscation threshold specified by network operators. Besides creating virtual nodes for reducing the popularity of nodes and links, EqualNet also generates virtual nodes to prevent adversaries from discovering the real nodes within the network. Such virtual nodes always remain with their corresponding real node; this way, adversaries are unable to discover which nodes are the real ones even after observing slightly different virtual topologies exposed at different times. This is in contrast to virtual nodes created for reducing the popularity of nodes and links only, which may be added to a virtual topology but do not necessarily need to be present in subsequent virtual topologies.

4) Network operators cannot leverage path tracing information to debug their networks. It is important for virtual topologies not only to be *secure* but also to maintain the *utility* of the information provided by path tracing tools. The fundamental problem with previous works is that they all rely on avoiding certain real nodes (i.e., the bottleneck nodes) from answering the tracing requests that traverse them (in order to protect them from adversaries). Existing solutions skip real nodes and reroute tracing requests to other nodes which handle and respond to the tracing requests intended for bottleneck nodes. To preserve the utility of the path tracing information, NetHide exposes secure virtual topologies while trying to skip as few real nodes as possible (being 1 their best case). However, in practice NetHide will often find a secure virtual topology only if multiple real nodes are skipped (as 1-hop neighboring nodes of the bottleneck node are likely to be as popular as the latter). In such a case, the use of NetHide can result in a significant utility loss in the path tracing information. Unlike NetHide, neither Trassare et al.'s solution nor LinkBait consider utility when generating their virtual topologies, rendering both solutions impractical. In the former all tracing packets are handled by a single node (the ingress router), whereas in the latter tracing packets are rerouted to a completely different path (the branch and bait links). Hence, these solutions not only prevent network operators from discovering failures in the bottleneck node and in any of its neighbor nodes and links, but also can provide misleading information to them.

EqualNet. Unlike previous work, EqualNet does not skip real nodes to conceal them from adversaries. This is crucial to be able to detect failures in any node in the network. By creating virtual nodes such that their IP addresses are in the same subnet as their corresponding real nodes, EqualNet ensures that tracing responses preserve the utility of path tracing information (while preventing adversaries from inferring any information that could allow them to distinguish between real and virtual nodes). This allows EqualNet to hide the popularity of nodes and links from adversaries, while giving enough information to in-network operators to debug their network.

Besides the aforementioned limitations, another problem of existing solutions is that none of them tested their obfuscation algorithms using real router-level network topologies for which the IP addresses are known. Unlike previous work, we performed our experiments using realistic network topologies along with their corresponding real IP addresses.

IV. Threat Model and Assumptions

We consider a layer-3 physical network managed by a single operator (e.g., an ISP), where all routers are SDN-capable. (In Section VI, we relax this assumption and consider a scenario where only some routers support SDN). Similarly to real-world SDN networks, the network's control plane is logically centralized and is realized through a cluster of SDN controllers (i.e., one main controller and several replicas of it to avoid scalability issues).

Threat model. The goal of adversaries is to conduct efficient and effective LFAs aimed to congest popular nodes and links – whose failure can negatively affect many network users – by injecting a large number of low-volume flows[6] indistinguishable from normal traffic. Coremelt [42] and Crossfire [31] are two representative research works that demonstrate the practicality of LFAs. As shown by these attacks, adversaries are not required to know any information about the capacity or load of routers and links. Adversaries can control many malicious hosts but they do *not* have any prior knowledge of the network topology. However, they can employ path tracing tools[7], such as traceroute, in order to obtain insights about the network topology as well as use state-of-the-art alias resolution techniques to check whether various router interfaces belong to the same router [40], [23], [33].

Assumptions. We assume adversaries do not know the network's subnet configuration information, such as network prefixes in Classless Inter-Domain Routing (CIDR). Moreover, we assume adversaries are bounded, i.e., they have limited resources to mount their attacks. Note that otherwise it would be impossible to defend against LFAs.

V. EqualNet

In this section, we present EqualNet. We start by introducing the notation we use throughout this paper along with EqualNet's main goals and the metrics used to measure the security and utility of its virtual topologies. Subsequently, we describe its main components in detail.

[6]A *flow* refers to a set of packets exchanged between a pair of source-destination IP addresses.

[7]If a MPLS-like protocol is used to route traffic within the network, the MPLS traceroute command [11] can be used, which achieves the same goal as traceroute but in a slightly different way.

TABLE I. EqualNet notations and metrics

Notation	Meaning
$\mathcal{G}^p$	Physical topology
$\mathcal{N}^p$	Physical node (router)
$\mathcal{L}^p$	Physical link
$\mathcal{G}$	Logical topology
$\mathcal{N}$	Logical node (IP address)
$\mathcal{L}$	Logical link
$\mathcal{G}'$	Virtual (logical) topology
$\mathcal{N}'$	Virtual (logical) node
$\mathcal{L}'$	Virtual (logical) link
fd^N	Node flow density
fd^L	Link flow density
$P_{s\to d}$	A forwarding path between s and d
$P^p_{s\to d}$	A forwarding path between s and d for a prefix p
T^p	A set of all forwarding trees for a prefix p
$leak(\mathcal{G})$	Topology leakage of $\mathcal{G}$
$sim(\mathcal{G}', \mathcal{G})$	Topology similarity of $\mathcal{G}'$ given $\mathcal{G}$
$util(\mathcal{G}', \mathcal{G})$	Topology utility of $\mathcal{G}'$ given $\mathcal{G}$

Notation. Table I summarizes the notation we use throughout this paper. We denote a physical topology by $\mathcal{G}^p = (\mathcal{N}^p, \mathcal{L}^p)$ where $\mathcal{N}^p$ and $\mathcal{L}^p$ refer to its routers and links, respectively. Note that an Autonomous System (AS) typically determines its forwarding behavior through Interior Gateway Protocols (IGPs) (e.g., OSPF) used to compute low-weight routing paths. Similarly, we denote a logical topology by $\mathcal{G} = (\mathcal{N}, \mathcal{L}, fd^{\mathcal{N}}, fd^{\mathcal{L}})$, where $\mathcal{N}$ and $\mathcal{L}$ refer to the logical nodes and logical links, respectively, and fd^N and fd^L correspond to the *flow density* of a node[8] and a link (i.e., the total number of unique flows that pass through it). Following the same method, we denote a virtual (logical) topology by $\mathcal{G}'$. A flow is denoted by $f = (s, d, a)$, where $s \in S$ is a set of sources, $d \in D$ is a set of destinations, and a is the amount of tracing flows (e.g., 240K) sent from s to d. Here, s and d represent nodes such as individual hosts or IP prefix blocks that send or receive tracing flows.

Goals and proposed metrics. Inspired by the cryptographic notion of indistinguishability, EqualNet aims to expose virtual topologies with equalized path tracing flow distribution over all network nodes and links. This way, adversaries who use path tracing tools (to retrieve network topology information) are unable to distinguish between popular and non-popular nodes by looking at the number of times each node appears in their tracing responses. One of the main challenges is how to achieve the latter while maintaining the utility of the information provided by path tracing tools. Thus, the goal of EqualNet is to generate virtual topologies that jointly maximize security and utility. Below, we describe which metrics we utilize to measure the level of security and utility of the virtual topologies EqualNet generates.

Topology leakage (security). We denote as *topology leakage* the difference in flow density between the node that receives the most tracing flows (i.e., the most popular node) and the one that receives the least tracing flows (i.e., the least popular node).

[8]More precisely, the node flow density refers to the sum of flow densities for all incoming links to the node.

Hence, the leakage of a given logical topology $\mathcal{G}$ is formally defined as:

$$leak(\mathcal{G}) = |\max(fd^N) - \min(fd^N)|$$

Ideally, to prevent adversaries from gaining any insights about the popularity of nodes and links, tracing flows should be uniformly distributed over all nodes and links. This way, all nodes would appear in tracing responses with equal probability, and hence the topology leakage would be nonexistent. In such a case, adversaries who use path tracing tools would not gain any advantage over blind (inefficient) LFAs executed without such tools.

Topology similarity (security). The topology similarity refers to how similar a given logical topology $\mathcal{G}$ and its corresponding virtual logical topology $\mathcal{G}'$ are at the graph level. The lower the topology similarity, the more random and complex the virtual topology is, which results in greater security. To measure the topology similarity, we compute the graph edit distance [22] between $\mathcal{G}$ and $\mathcal{G}'$. This metric reports on the minimum number of operations required to transform $\mathcal{G}'$ into $\mathcal{G}$. Formally, the topology similarity between $\mathcal{G}$ and $\mathcal{G}'$ is defined by:

$$sim(\mathcal{G}', \mathcal{G}) = 1 - \frac{GED(\mathcal{G}', \mathcal{G})}{|\mathcal{G}'| + |\mathcal{G}|},$$

where $GED(\mathcal{G}', \mathcal{G})$ denotes the graph edit distance between $\mathcal{G}$ and $\mathcal{G}'$. $|\mathcal{G}|$ and $|\mathcal{G}'|$ correspond to the sum of the number of nodes and links in the real logical topology and the virtual logical topology, respectively.

Topology visibility (utility). This metric reflects how well virtual topologies can assist network operators in finding network failures. The better the subnet-level visibility, the more chances of network operators being able to quickly fix such failures. We denote as $P_{s\to d}$ a tracing path (i.e., a sequence of IP addresses corresponding to the nodes traversed) from source s to destination d. $P^p_{s\to d}$ refers to the same tracing path but this one only contains the subnet-level information, while T^p corresponds to the subnet-level network's forwarding tree for a given prefix p. Consider a scenario where (external) network operators observe a failure at the i-th hop (e.g., in the 3rd node) in a tracing path $P_{s\to d}$. The probability of network operators (within the AS) being able to accurately locate the network failure (based on the tracing information they receive from external network operators) depends on the number of forwarding paths $numFP$ with length $i-1$ which has identical subnet-level information as $P_{s\to d}$:

$$numFP(T^p, P^p_{s\to d}, i) = |T^p[0:i-1] \cap P^p_{s\to d}[0:i-1]|,$$

where $[0:i-1]$ denotes the sequence of IP addresses before the failure hop i. Therefore, the utility of a virtual topology $\mathcal{G}'$ given a logical topology $\mathcal{G}$ is defined as[9]:

$$util(\mathcal{G}', \mathcal{G}) = avg\left(\sum_{(s,d)\in T^p, s\neq d} \sum_{i=2}^{|P^p_{s\to d}|} \frac{1}{numFP(T^p, P^p_{s\to d}, i)}\right)$$

[9]Note that path tracing information does not help detecting failures in the first hop (i=1) since in that case network operators do not obtain any information when using path tracing tools.

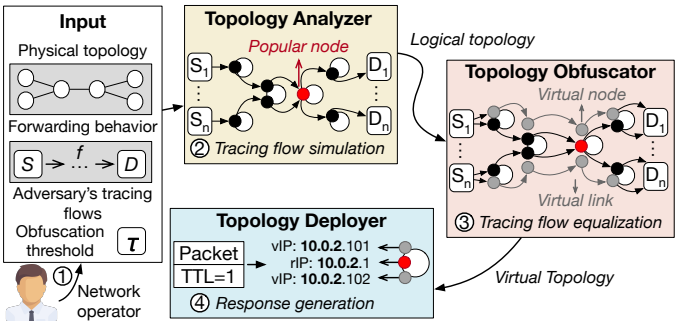


Fig. 2. EqualNet's system overview. 1) Network operators provide the physical topology with its forwarding behavior, the obfuscation threshold and the adversary's tracing flows. 2) The topology analyzer computes the logical topology and the tracing flow distribution from the adversary's perspective. 3) The topology obfuscator generates a virtual topology that equalizes the tracing flow distribution. 4) The topology deployer continuously monitors tracing flows and generates tracing responses with real or virtual IP addresses.

Overview. Given an unprotected (i.e., with imbalanced tracing flow distributions) network topology, the goal of EqualNet is to generate a virtual topology that significantly reduces the topology leakage in the network. As a first step, EqualNet analyzes the view of the network topology from the adversary's perspective and computes the topology leakage. Subsequently, EqualNet creates virtual nodes (i.e., instructs real nodes to reply to tracing packets with IP addresses different to theirs) for two different purposes: (i) to conceal real nodes from adversaries long-term and (ii) to decrease the popularity of the most observed nodes and links. The latter is crucial to decrease the topology leakage; we design EqualNet such that network operators can specify their desired level of obfuscation depending on the level of security they need and the overhead they are willing to tolerate. Meanwhile, EqualNet assigns virtual nodes IP addresses within the same subnet as the one from the real node from which they originate, maintaining the utility of path tracing information. Whenever a node appears too frequently in the adversary's tracing responses compared to the other nodes, EqualNet automatically adjusts the exposed virtual topology to decrease the topology leakage based on the level of security network operators wish to have.

Example. Consider again the running example described in the previous section. Figure 1e illustrates the way EqualNet could conceal the bottleneck link E1-G1 from adversaries. Note that this is just a simple example where EqualNet creates an additional virtual node in nodes A, D, E, G, I and J. (However, in practice EqualNet can create more virtual nodes in each physical node to reduce the topology leakage). EqualNet exposes a virtual topology containing several virtual nodes (denoted as $V1$-$V6$), whose network IP prefixes are the same as those in the physical nodes where they are created (to preserve subnet information), and their host IP addresses are selected at random (to provide security). Hence, when EqualNet is used, the adversary would obtain a traceroute response containing either A1 → D1 → E3 → G1 → I1 → J1 or V1 → V2 → V3 → V4 → V5 → V6, where both responses are *equally likely* (and fully disjoint). The latter is crucial because in such a case adversaries can neither learn which nodes and links are popular nor infer real topologies among different topology instances.

We propose two variants of EqualNet. The first one estimates the adversary's view of the network topology offline based on information provided by the network operators. Then, it uses this information along with the obfuscation threshold (also specified by the network operators) to compute (offline) the number of virtual nodes required to guarantee that the topology leakage is below the obfuscation threshold. Afterwards, EqualNet exposes a virtual topology that attempts to reduce the topology leakage in a best-effort manner using solely the number of nodes created offline. This variant of EqualNet is suitable for network operators who do not want to create too many virtual nodes. Yet, it requires network operators to possess information about the adversary's behavior (which may be unavailable or inaccurate). In contrast, the second variant operates *fully online* and performs the obfuscation tasks *on-the-fly* (i.e., as tracing packets arrive to the network) without requiring the definition of the adversary's behavior. Following the procedure previously described, the second variant of EqualNet first computes the topology leakage and then attempts to decrease it. Here, rather than providing the desired topology leakage reduction, network operators are required to provide the maximum difference in number of tracing flows they can tolerate between any pair of nodes. In this case, the goal of EqualNet is to guarantee that the topology leakage does not exceed the tolerated number.

As shown in Figure 2, EqualNet comprises three main components: (i) the topology analyzer (Section V-A), (ii) the topology obfuscator (Section V-B) and (iii) the topology deployer (Section V-C). In the next subsections, we will introduce the main components of EqualNet focusing on how these work when applied to the first variant of EqualNet. At the end of this section, we will present the second variant of EqualNet.

A. Topology Analyzer

The topology analyzer starts by inspecting a given (unprotected) network topology in order to infer the network topology view observed by an adversary who uses path tracing tools. Then it analyzes the distribution of path tracing flows to find the most popular nodes and links.

Building the adversary's logical topology. Initially, the topology analyzer conducts a series of simulations offline to find the network topology the adversary would observe from the tracing responses it obtains. To that end, the topology analyzer requires network operators to provide the network topology, its forwarding behavior and the adversary's tracing flows. For the latter, one option is to let network operators specify the adversary's behavior themselves (e.g., based on traces from past attacks). Alternatively, the topology analyzer can use some default configuration, e.g., assume a worst case scenario where adversary's flows are sent from all ingress routers. (In Section V-D, we introduce the second variant of EqualNet, which does not require network operators to provide the adversary's tracing behavior as input). In the rest of this section, we assume that network operators have valuable information about past attacks which they can use to model the adversary's tracing behavior.

The topology analyzer proceeds with its (offline) simulations as follows: it takes the *adversary's tracing flows*, sends them to the corresponding destination nodes and collects the produced tracing responses. The adversary's tracing flows are given as a triplet containing the source and destination IP prefix blocks (or wildcard rules) along with their corresponding flow amounts. For example, a triplet (10.1.0.0/16, 10.2.0.0/16, 240K) mimics the behavior of an adversary who uses source nodes in 10.1.0.0/16 to send 240K different tracing flows to destination nodes in 10.2.0.0/16. Figure 3 shows the view of the network topology obtained by an adversary who sends 240K tracing flows between all source-destination pairs in the network.

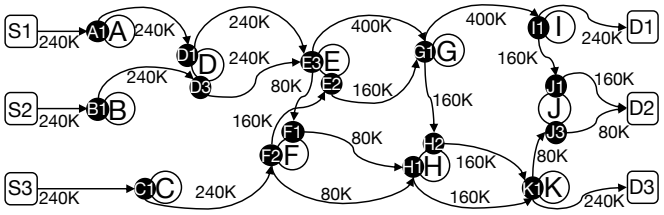


Fig. 3. Example of adversary's logical topology view built upon the physical topology in Fig. 1a. Rectangles denote source and destination nodes. The white and black circles refer to physical nodes (not seen by path tracing tools) and logical nodes visited by flows, respectively. The displayed numbers on the links denote the sum of the expected flow count (i.e., their flow density) per logical link.

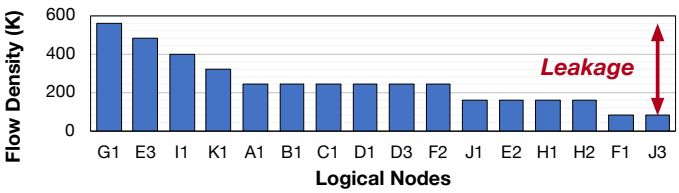


Fig. 4. Measured tracing flow distribution in our running example. The red arrow refers to the current topology leakage in the network.

Algorithm 1. Next, we introduce the graph-based algorithm used to construct the network's logical topology from a given physical topology and adversary's tracing behavior. Given a physical topology, its forwarding behavior and the adversary's tracing flows, Algorithm 1 picks the first adversary's tracing flow, which specifies how many tracing flows a are sent from the source host s to the destination host d (line 2). Then it computes the physical path $path_{s \to d}$, which contains all physical nodes along the shortest path from the source s to destination d (line 3). Starting from the source, a tracing flow visits each i-th physical node n_i^p along the physical path $path_{s \to d}$ (lines 4 to 5). For each visited physical node, the algorithm obtains a tracing response with the IP address of the router's ingress interface (lines 6 to 7). Then, it creates a logical link l_i from the previous node (the last visited one) n_{i-1} to the current node n_i (line 8). If the logical node and link do not (yet) exist in the logical topology, they are added to it (lines 9 to 12). The algorithm also adds the number of tracing flows a to the flow density of the corresponding link and node, respectively (denoted as $fd^{\mathcal{L}}(l_i)$ and $fd^{\mathcal{N}}(n_i)$) (lines 13 to 14). This process is repeated for all physical nodes along this specific path then for all other adversary's tracing flows. Finally, the algorithm returns the logical topology $\mathcal{G}$ which reflects the adversary's view of the network (line 15).

Algorithm 1 Logical topology generation

Require:
Physical topology $\mathcal{G}^p = (\mathcal{N}^p, \mathcal{L}^p)$
Adversary's tracing flow set $F = \{f_1, f_2, ..., f_x\}$
$f = (s, d, a), s \in S, d \in D, a \in \mathbb{N}$
Ensure:
Logical topology $\mathcal{G} = (\mathcal{N}, \mathcal{L}, fd^N, fd^L)$
$\mathcal{N} \leftarrow \emptyset, \mathcal{L} \leftarrow \emptyset$ ▷ Initialization
for $f \in F$, where $f = (s, d, a)$ **do**
  $path_{s \to d} \leftarrow$ GETFORWARDINGPATH$(\mathcal{G}^p, s, d)$
  $n_0 \leftarrow s$ ▷ Initialize the current logical node
  for $i \leftarrow 1$ to $path_{s \to d}$.length **do**
    $n_i^p \leftarrow path_{s \to d}[i]$
    $n_i \leftarrow$ GETINTFIPADDR$(\mathcal{G}^p, n_{i-1}^p, n_i^p)$
    $l_i \leftarrow (n_{i-1}, n_i)$ ▷ Make a logical node and link
    if $n_i \notin \mathcal{N}$ **then** ▷ If n_i does not exist, add it
      $\mathcal{N} \leftarrow \mathcal{N} \cup \{n_i\}$
    if $l_i \notin L$ **then** ▷ If l_i does not exist, add it
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{l_i\}$
    $fd^{\mathcal{N}}(n_i) \leftarrow fd^{\mathcal{N}}(n_i) + a$ ▷ Update node flow density
    $fd^{\mathcal{L}}(l_i) \leftarrow fd^{\mathcal{L}}(l_i) + a$ ▷ Update link flow density
$\mathcal{G} \leftarrow (\mathcal{N}, \mathcal{L}, fd^{\mathcal{N}}, fd^{\mathcal{L}})$

Identifying popular nodes and links. Besides revealing the logical topology, tracing responses can provide the adversary with insights about the *tracing flow distribution* in the network. Unfortunately, regardless of their size, networks typically contain a few nodes that handle most of the traffic, provoking an imbalanced path tracing distribution over nodes and links [30]. Consider the network topology shown in Figure 3 and its corresponding tracing flow distribution (see Figure 4). Due to the imbalanced tracing flow distribution, there exists a significant difference in flow density between the most popular logical node (i.e., $G1$) and the least popular ones (i.e., $F1$ and $J3$). As a result, the current network topology has a high topology leakage, meaning that valuable information about the network characteristics is disclosed to adversaries.

EqualNet allows network operators to define an obfuscation threshold to limit the maximum topology leakage permitted in the network. This provides greater flexibility since it allows adjusting EqualNet according to the needs of network operators. Here, the obfuscation threshold refers to the *topology leakage reduction* (measured in %) network operators wish to achieve with respect to the topology leakage in the original network topology. The higher the topology leakage reduction is, the more equalized the tracing flow distribution will be, and thus the more difficult will be for adversaries to gain insights about the popularity of nodes and links. However, as shown in Section VI, this typically also comes at the cost of generating more virtual nodes, which is undesirable. Therefore, there is the need to find a suitable balance between the chosen obfuscation threshold and the number of virtual nodes generated (see Section V-B for more details).

B. Topology Obfuscator

Estimating the number of virtual nodes to obfuscate the network topology. The topology obfuscator equalizes the path tracing distributions over nodes and links in a best-effort manner while keeping the number of created virtual nodes relatively small. Suppose the topology leakage in the original network topology is 600 flows (as in Figure 4), and that network operators wish a 80% topology leakage reduction. In such a case, the goal of EqualNet is to lower the topology leakage from 600 to (at most) 120 flows.

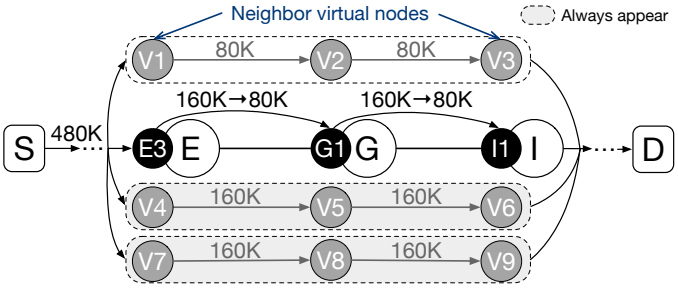


Fig. 5. Procedure through which the obfuscation algorithm equalizes the tracing flow density of a popular node (G1) by adding virtual nodes (V1, V2, and V3) and a fixed set of virtual nodes (which always appear in the exposed virtual topologies) to hide real nodes from adversaries (V4-V9).

Independently of the topology leakage in the original logical topology, the first step for the topology obfuscator is always to generate a small, fixed number of (guard) virtual nodes in all real nodes in order to protect them from adversaries who can observe multiple, slightly different virtual topologies exposed over time. For example, all real nodes in Figure 5 contain two virtual nodes each for this purpose (V4 and V7 in node E, V5 and V8 in node G, and V6 and V9 in node I). Essentially, the intuition behind this method is to use those virtual nodes to produce a sort of anonymity set – whose size does not need to be large – to avoid adversaries from inferring which nodes are real long-term. Therefore, those virtual nodes must always be present in any virtual topology exposed by EqualNet.

Afterwards, for each real node, EqualNet equalizes the tracing flow distribution among the real nodes and the existing (guard) virtual nodes, and computes the new topology leakage. If the topology leakage is still higher than the target (i.e., 120 flows), EqualNet creates additional virtual nodes (in Figure 5 the topology obfuscator creates V2) to further split tracing flows among virtual nodes, causing the popularity of certain nodes and links to decrease even further. To maintain consistent logical paths, all subsequent tracing packets within the same flow get the same set of logical and virtual nodes as the first tracing packet.

There exists an important aspect to be considered when adding virtual nodes to the virtual topology. From a security point of view, it is important for virtual logical nodes not to be connected to real logical nodes, since there can only be one node connected to a router's interface at a time and the latter are already connected to other (real) logical nodes. Otherwise, this could disclose to adversaries that two IP addresses belong to the same node (see the discussion on alias resolution in Section VII for more details). To satisfy this requirement, whenever the topology obfuscator creates a virtual node in a real node (e.g., V2 in real node G), it also creates a virtual node in each of its neighbor nodes (e.g., V1 in node E and V3 in node I) and then connects them so that they form fully disjoint paths from those containing all real logical nodes (see Figure 5). As a result, adversaries who use path tracing tools can either obtain (i) E3 $\rightarrow$ $G1$ $\rightarrow$ I1, (ii) V4 $\rightarrow$ $V5$ $\rightarrow$ V6, (iii) V7 $\rightarrow$ $V8$ $\rightarrow$ V9 or (iv) V1 $\rightarrow$ $V2$ $\rightarrow$ V3.

The procedure is then repeated until the topology obfuscator generates a virtual topology whose topology leakage is equal or below the target previously defined (e.g., 120 flows in the previous example) or until there are no IP addresses available in (at least) one of the subnets. Once this is finished, the topology obfuscator outputs the number of virtual nodes that need to be created considering the previously defined obfuscation threshold. These are the virtual nodes available to the topology deployer to obfuscate the network topology online (see next subsection). For more details on how the offline obfuscation works, we refer the reader to Appendix A.

C. Topology Deployer

The last step for EqualNet is to deploy the previously generated virtual topology in the network leveraging SDN. The topology deployer generates flow-rules and instructs routers to reply to tracing packets either with any of their real IP address or with a fake one. The deployment of the virtual topology can be divided into two steps: (i) the realization of virtual nodes using SDN and (ii) the assignation of IP addresses to virtual nodes. Next, we describe both steps in detail.

Realizing virtual nodes. Leveraging the lack of security mechanisms to preserve the integrity and authenticity of tracing packets, the topology deployer modifies the source IP address field in tracing responses in order to create virtual nodes and links in the logical topology. This requires some (minor) configuration changes in the network that only needs to be done once. As SDN-enabled routers do not decrease the TTL value by default [18], the topology deployer installs a permanent flow-rule in all routers with the action *Decrement_IP_TTL* so that they decrement the TTL field by 1 in the tracing packets they receive. It is important to note that only 1 flow rule is required for this in each router and that EqualNet does not require routers to keep any other state. In addition, the SDN controller sends a *Set_Async_Config* OpenFlow packet to each router to enable the *Invalid_TTL* flag. By doing so, every time a router receives a tracing packet for which the TTL is invalid, the router automatically encapsulates it within a *Packet_In* OpenFlow packet and forwards it to the SDN controller.

The controller keeps an *obfuscation map* that is constantly being updated. It comprises a 3-tuple: {*flow ID*, *router interface* and *response IP address*}. The *flow ID* refers to the source and destination IP address prefix, the *interface* denotes the router interface where the tracing packet is received and the *response IP address* corresponds to the IP address EqualNet assigns to the tracing response. Figure 6 shows an example of tracing response generation. Suppose the interface $G1$ receives a tracing packet (belonging to a new tracing flow). The first step is for router G to send it to the controller[10] via an OpenFlow *Packet_In* packet. Then, the controller chooses a valid response IP from the obfuscation map and sends an OpenFlow *Packet_Out* packet to router G containing a crafted tracing response with the virtual IP address $V2$ (see Figure 6).

Assigning IP addresses to virtual nodes. To assign IP addresses to newly-generated virtual nodes, the topology deployer could follow various strategies. One possibility would be to allocate random IP addresses to virtual nodes. However, the use of random IP addresses in tracing responses would be noticeable to adversaries, which could then leverage this information to distinguish between real and virtual nodes. Moreover, this approach would not maintain the utility of the path tracing information. Instead, the topology deployer opts for assigning virtual nodes randomly-chosen IP addresses *within the same subnet* as their real logical nodes. To avoid using already assigned IP addresses, the topology deployer periodically interacts with the SDN controller and retrieves the list of assigned IP addresses from its topology discovery services.

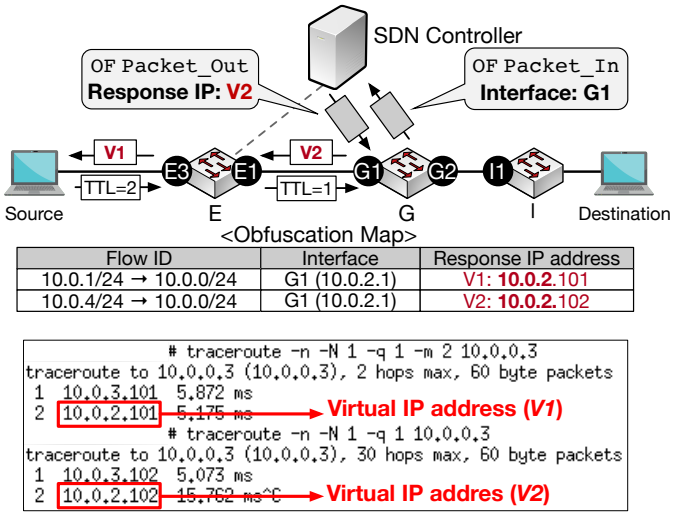


Fig. 6. The example scenario (top) and the result (bottom) of response generation to realize virtual nodes V1 and V2 in the path tracing tool when detecting a tracing packet whose TTL is 1 at interface G1.

Obfuscating the network topology long term. Once the virtual topology is exposed, the topology deployer continuously monitors all incoming tracing flows. Every time a node receives an expired tracing packet (with TTL=0) in one of its interfaces, it notifies the topology deployer and the topology leakage is re-computed. If the topology obfuscator detects that the topology leakage is above the tolerated value (based on the defined obfuscation threshold), it makes the necessary changes to the exposed virtual topology to decrease the topology leakage while creating as few virtual nodes as possible. (Recall that the maximum number of virtual nodes that can be produced is determined by the output of the topology obfuscator). Otherwise, the topology deployer balances the incoming tracing packets among existing real and virtual nodes.

The addition or removal of *virtual nodes* to/from the virtual topology (to decrease its topology leakage) does not leak any useful information to adversaries. This is because EqualNet adds a fixed number of virtual nodes – which are always present in all virtual topologies exposed – to permanently protect all real nodes in the network. We also argue that it is not a problem if a few real nodes are added or removed since adversaries are unable to know how many of such nodes are real nodes and how many of them are virtual ones. While in our experiments we created the same number of virtual (guard) nodes for each of the real nodes, in practice

[10] In any SDN network, tracing packets are processed by the control plane and hence always need to be forwarded to the SDN controller.

network operators could decide to generate a different number of permanent virtual nodes per node. This approach would make it even harder for adversaries to learn any information about the number of real and virtual nodes being added or removed.

D. Second variant of EqualNet

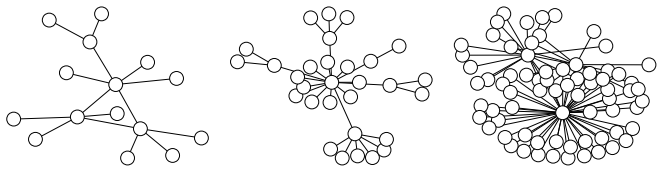
In the previous subsections, we presented the main components of EqualNet and showed how they are used by the first variant of EqualNet to hinder adversaries from knowing which are the most popular nodes and links. Next, we introduce EqualNet second variant.

Unlike the first variant, the second variant of EqualNet equalizes the tracing flow distributions on-the-fly (i.e., as tracing packets are sent to the network) and hence is fully online. This means that network operators who do not have (or do not want to use) past attack data can also use EqualNet. Network operators only need to provide the network topology as well as the desired obfuscation threshold to EqualNet, which then uses this information along with the received tracing packets to compute the current topology leakage and reduce it to a large extent. In this case, the obfuscation threshold refers to the maximum difference in number of tracing flows between any pair of nodes allowed in the network. From this point onward, EqualNet follows a similar obfuscation procedure as the one shown in the first variant. The only difference compared to the first variant is that now EqualNet is not limited to using solely the number of virtual nodes generated offline. As a result, EqualNet can create as many virtual nodes as needed to keep the topology leakage under the maximum tolerated value for as long as there are available IP addresses in the subnets of the real nodes.

VI. Implementation and evaluation

Implementation. To evaluate the feasibility of our solution, we developed a full prototype of EqualNet using various SDN applications (2K+ lines of code) running atop Ryu v4.29 [8], a widely known open-source SDN controller. We implemented a separate SDN application for each of the three modules comprising EqualNet, namely (i) *topology analyzer*, (ii) *topology obfuscator* and (iii) *topology deployer*. In addition, EqualNet includes two additional SDN applications – which we name *tracing flow forwarding manager* and *alias resolution handler*. The former is used to maintain information about the tracing flows sent to routers and to install rules for forwarding unexpired tracing packets to a destination, whereas the latter intercepts tracing packets that target virtual nodes and generates valid responses to those packets in order to prevent adversaries from fingerprinting routers. To evaluate our obfuscation algorithms, we leverage the Mininet [35] network simulator, which runs Open vSwitch v2.5.5 [7] and supports the OpenFlow v1.3 specification. We conducted our experiments in an Intel Xeon Silver 4210 CPU 2.20Ghz with 64GB RAM.

Topology dataset. To assess the effectiveness of our algorithms, we used the CAIDA Internet Topology Data Kit (ITDK) dataset [10], which comprises a set of router-level network topologies. As those datasets incorporate real IP addresses observed in the wild, they can be used for simulating adversaries who use path tracing tools to discover popular nodes within an AS. Given that this dataset contains information about millions of routers without distinguishing inter- and intra-AS connections, we adjusted it to be tailored for our purpose (intra-AS links). Then, we selected the following three networks: (i) AS 13576 (15 physical nodes and 30 links), (ii) AS 35132 (30 physical nodes and 60 links), and (iii) AS 35575 (71 physical nodes and 140 links) in order to evaluate the suitability of EqualNet when applied to small, medium, and large-sized networks (see Figure 7). In the rest of this section, we consider that each node sends 1K tracing flows to each of the remaining nodes in the network to construct the initial logical topology. Similarly, for evaluations 1-5, each host sends 1K tracing flows to each of the remaining hosts [12].



(a) AS 13576 (small) (b) AS 35132 (medium) (c) AS 35575 (large)

Fig. 7. Router-level topology data from CAIDA ITDK [10].

1) Effectiveness of tracing flow equalization. We tested the obfuscation algorithms of both variants of EqualNet with several obfuscation thresholds[11]. Figure 8 shows the inverse complementary cumulative distribution functions (CCDF) of flow densities measured in the three network topologies. An ideal result would be that tracing flow distributions in nodes and links exhibit close to horizontally flattened patterns (i.e., the τ_I=80% case is the best one). As expected, the selection of strict obfuscation thresholds leads to very equalized path tracing flow distributions. However, we also observe that in both variants, the tracing flow distributions in the three network topologies are largely equalized even when the least strict obfuscation thresholds are used (i.e., τ_I=20% and τ_{II}=0.5K). For example, about top 60% nodes in small and medium ASes and top 90% of nodes in the large AS exhibit similar flow density values with the least strict obfuscation threshold τ_{II}=5K (see dashed green lines). Overall, this experiment shows that the obfuscation algorithms used by EqualNet to equalize the tracing flow distributions are very effective at hiding the popularity of nodes and links in the network.

2) Topology leakage reduction vs. required virtual nodes. We analyzed the number of virtual nodes that need to be created for achieving different levels of obfuscation. As shown by Figure 9, the stricter the obfuscation threshold is, the more virtual nodes are needed. For example, in EqualNet variant I, using the strictest obfuscation threshold (τ_I=80%) in the large AS requires around 75 virtual nodes in each router, which is 9x larger than when the loosest threshold (τ_I=20%) is applied. However, it is interesting to see that only with the loosest threshold EqualNet can equalize the tracing flow distribution of the large AS considerably while creating a reasonably small number of virtual nodes (around 9 virtual

[11] In the first variant, the obfuscation threshold denoted by τ_I, refers to the desired topology leakage reduction. In contrast, in the second variant, the obfuscation threshold denoted by τ_{II}, refers to the maximum difference in number of tracing flows between any pair of nodes allowed in the network.

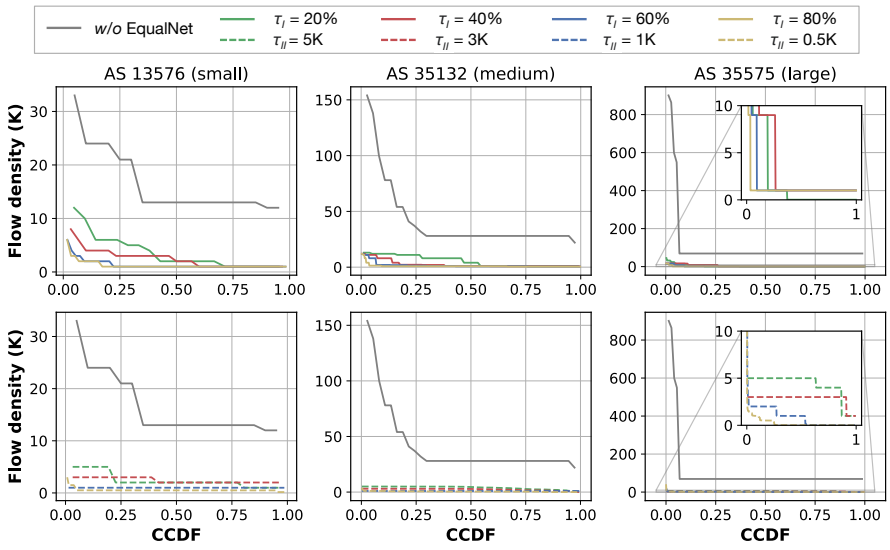
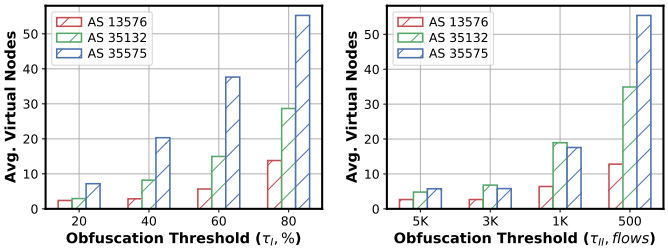


Fig. 8. Equalized flow density distributions for each of the topologies for the variant I and II using diverse thresholds τ_I and τ_{II}, respectively. Note that the more horizontally flattened, it implies that the more flow density distributions are equalized.



(a) Variant I (b) Variant II

Fig. 9. Average number of virtual nodes that need to be created in each real node for different obfuscation thresholds.

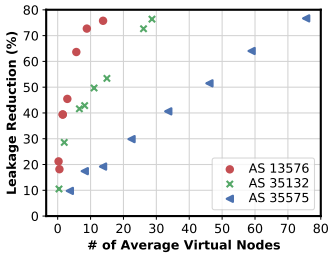


Fig. 10. Trade-off between topology leakage reduction and the number of average virtual nodes required for a single router.

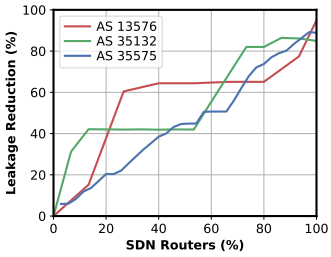


Fig. 11. Effectiveness of obfuscation under the partial SDN deployment at important network locations.

nodes in each real node which results in a total number of virtual nodes of around 600). Moreover, we performed some experiments to investigate the relationship between the number of virtual nodes generated and the achieved topology leakage reduction for the three network topologies. Figure 10 shows that for the large AS the number of virtual nodes required grows linearly as the topology leakage reduction increases. However, we observed that this does not always hold in all network topologies. For example, using 14 virtual nodes in each router in the small AS achieves a 76% topology leakage reduction; note that this only improves the topology leakage reduction by just 5% compared to the case when 9 virtual nodes are used in each router. This experiment shows that there is a need to find an optimal balance between the topology leakage reduction and the number of virtual nodes such that the topology leakage is reduced significantly while keeping the number of created virtual nodes as small as possible. As our results reveal, there might be a point where EqualNet requires to add a large number of virtual nodes to lower the topology leakage only very slightly. We argue that network operators can use this to determine the optimal number of virtual nodes that need to be created.

3) Partial SDN deployment. We analyzed the feasibility of deploying EqualNet in networks where only a subset of routers support SDN. In such a case, only SDN-enabled routers can obfuscate their interfaces; other routers are forced to expose their real interfaces to tracing flows. We considered the case in which network operators decide to use SDN in centralized core routers. To that end, we first computed the betweenness centrality [21] for all physical nodes and incrementally selected the nodes with the highest centrality as SDN routers. Then we measured the topology leakage reduction EqualNet can achieve when varying the percentage of SDN-capable devices in the network. As we expect that such central routers will receive many more tracing flows than others, we used the strictest obfuscation threshold τ_{II}=0.5K. Figure 11 shows that by just having 20% of SDN-capable routers, EqualNet was able to reduce the topology leakage by 40% in the small and medium ASes. In contrast, in the case of the large AS, the results indicate that there exists a linear relation between the topology leakage reduction and the number of SDN-capable routers in the network. Interestingly, these results also show that having SDN in more than 80% and 50% of their routers for the small and medium ASes does not lead to significant improvements in terms of leakage reduction, making this deployment strategy (80% and 50%, respectively) the most optimal. As these high-centrality nodes received many tracing flows, EqualNet was able to efficiently apply its obfuscation algorithms in order to lower the topology leakage.

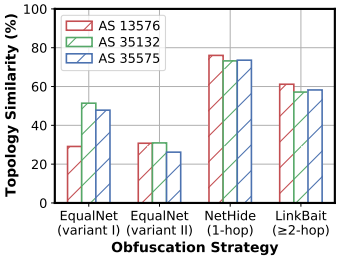


Fig. 12. Comparison of EqualNet's variants and previous network obfuscation solutions in terms of topology similarity (the lower, the more secure the virtual topology is).

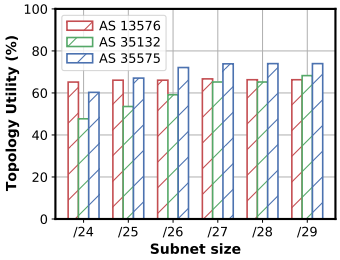
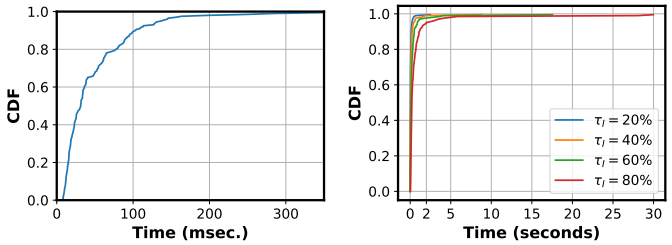


Fig. 13. Measured utility of EqualNet for different subnet sizes.



(a) Time for logical topology generation.

(b) Time for offline obfuscation.

Fig. 14. The scalability of EqualNet's algorithms.

Overall, our experiments showed that EqualNet can also significantly reduce the topology leakage in networks where only some of its routers support SDN.

4) Topology similarity. To assess the security of EqualNet, we measured the topology similarity of virtual topologies generated by EqualNet, and compared them with those produced by existing network obfuscation solutions. As none of the state-of-the-art topology obfuscation defenses made their obfuscation algorithms publicly available, we implemented them ourselves. We used moderate obfuscation thresholds (τ_I=40% and τ_{II}=3K) in both variants. Figure 12 shows a comparison between existing solutions and EqualNet's variants in terms of the topology similarity they offer. Our results show that EqualNet is able to produce the most dissimilar virtual topologies (in the best case EqualNet achieved a 76% similarity reduction). This stems from the fact that EqualNet expands the topology space by adding virtual nodes and links. This is in contrast to previous solutions, whose virtual topologies did not experience a significant similarity reduction. To illustrate this, consider the 2-hop obfuscation strategy followed by LinkBait, which exhibited (at best) 58% similarity reduction. For a fair comparison with EqualNet and Nethide, recall that, unlike the previous, LinkBait does not intend to provide utility in the virtual topologies it generates; without this requirement we note that it is easier to expose dissimilar virtual topologies.

5) Topology utility. We analyzed the topology utility of the virtual topologies created by EqualNet using the utility metric we introduced in Section V. As there is no available dataset for which the IP prefixes are known (this information is kept confidential), we simulated subnet-level traces (e.g., 10.0.2/24 $\rightarrow$ 10.0.3/24) from the CAIDA dataset. To that end, we considered a realistic range of prefixes that can be employed for a single node or link (e.g., from /24 to /29). Figure 13 shows that EqualNet could provide 60% utility in average for all network prefixes. The case when EqualNet did not achieve high utility (48% in AS 35132 for /24 prefix) is because of the way the IP assignment is done. In that AS, there are several physical nodes with IP addresses which, under various possible prefixes, would belong to the same subnet.

6) Scalability verification. We also evaluated the scalability of Algorithms 1 and 2. For this, we randomly selected 200 ASes from the CAIDA dataset whose size spans over small (61 nodes) to large networks (206 nodes). For each of the 200 ASes, we measured the (i) time it takes to generate logical topologies (Algorithm 1) and (ii) the time it takes to estimate virtual nodes needed offline (Algorithm 2). Figure 14 shows the distributions of measured execution time for Algorithm 1 and 2. Our results indicate that logical topology generation (Algorithm 1) takes less than 100 milliseconds in most ASes (around 80%) whose size is less than 150 nodes. For large-sized networks (206 nodes), we observe that this can take up to 356 milliseconds. Similarly, estimating the number of virtual nodes needed (Algorithm 2) takes less than 2 seconds in most cases (around 95%) regardless of the chosen value for topology leakage reduction. If a stricter obfuscation threshold is chosen, the time it takes for the algorithm to be completed will increase (e.g., 30 seconds in τ_I=80%). Given these results, re-running the algorithms (e.g., when the physical topology changes significantly) will not impose a significant overhead.

7) Latency. We examined the RTT distributions of various routers after sending them many tracing packets with and without EqualNet (see Figure 16). By comparing the RTT distributions with and without EqualNet, we observed that EqualNet does not introduce a noticeable delay (maintaining the utility of the RTT values included in tracing responses).

VII. Security Analysis

In a nutshell, the security of EqualNet is strongly related to three fundamental questions: Q1) *Can adversaries distinguish between real nodes?* Q2) *Can adversaries distinguish between real and virtual nodes?* Q3) *Can adversaries infer whether two IP addresses belong to the same subnet?*

Next, we analyze the security properties offered by EqualNet, highlighting the mechanisms it has to defend against a broad range of security attacks. Due to lack of space in this paper, we discuss how EqualNet defends against subnet inference attacks in Appendix B and elaborate on possible lines of work to further improve EqualNet in Appendix C.

Resistance to RTT-based fingerprinting attacks. Adversaries could attempt to distinguish the real nodes in the network from the RTT information in the tracing responses. This could allow them to know which virtual IP addresses belong to each real node, and ultimately retrieve the popularity of all real nodes and links in the network. To evaluate the resistance of EqualNet to such attacks, we conducted several experiments using a hardware testbed with the purpose of analyzing whether RTTs inside tracing responses can (i) act as a router's fingerprint (ii) reveal that EqualNet is being used and (iii) be used to distinguish between real and virtual nodes.

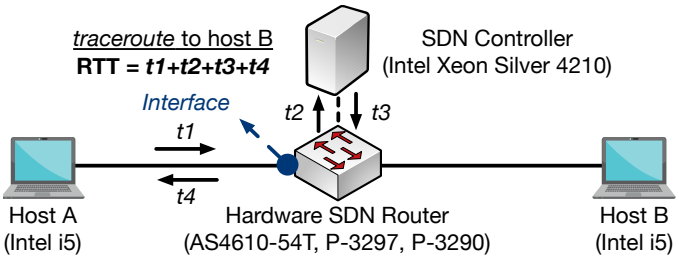


Fig. 15. The SDN hardware testbed and RTT measurement scenario.

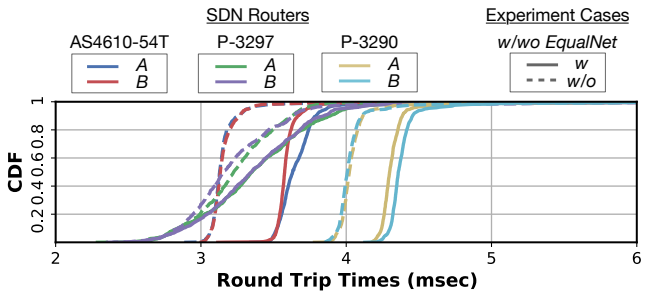


Fig. 16. CDF of RTTs measured in 6 SDN routers. (*w*: with EqualNet, *w/o*: without EqualNet)

Figure 15 shows the network setup we used. Our hardware testbed comprises 6 SDN routers from 3 different vendors: 2 EdgeCore AS4610-54T, 2 Pica P-3297 and 2 P-3290 devices. To prove the security of EqualNet, we considered a *worst-case scenario* where the adversary (i.e., Host A) is able to obtain tracing packets with *noise-free RTT measurements*. (Note that in practice these measurements will be heavily affected by noise and thus it will be even harder for adversaries to distinguish nodes based on RTT measurements). In particular, we selected 1 hardware router each time and placed it between Host A and Host B. For each of the 6 hardware routers, we then sent 1000 traceroute packets from Host A to Host B, and computed the average of the RTTs values obtained in the traceroute responses (see Figure 16).

Can RTTs act as a router's fingerprint? The dashed lines in Figure 16 denote the obtained RTT distributions for each of these devices when EqualNet is *not* used. While routers from the same vendor tend to exhibit a similar RTT distribution, no single router has a sufficiently different RTT distribution to be uniquely and precisely identified. We would like to stress that in any real-world network, the task of identifying routers from their RTTs will be even more difficult for various reasons. First, the propagation time in the channel is likely to be the dominant factor in the RTTs (instead of the processing time) as tracing packets have to traverse multiple nodes back and forth. As a result, the propagation delay of tracing packets is typically in the order of several milliseconds and can have a significant variance. Second, it is important to note that ASes typically contain a large number of routers, making it very difficult for adversaries to distinguish each of them. Because of this, we argue that it is hard for adversaries to distinguish the real nodes in realistic environments by looking at the RTT values within tracing responses.

Can RTTs reveal that EqualNet is being used? The straight lines in Figure 16 denote the obtained RTT distribution for each of these devices when EqualNet is used[12]. By comparing the RTT obtained with and without EqualNet, we can observe that EqualNet comes with a small and stable (compared to the RTTs obtained when EqualNet is not used) latency increase of less than 1 millisecond. While it is not our goal to hide the fact that EqualNet is being used in the network, due to the reasons previously explained it will be difficult for adversaries to learn such information from the RTTs they obtain.

Can RTTs be used to distinguish between real and virtual nodes? For every tracing packet sent to the network, EqualNet re-computes the topology leakage, selects a suitable node (i.e., the IP address) to respond and, in case no suitable node exists, generates a new virtual node. Among them, the most computationally intensive operation (and hence the one that influences latency the most) is the topology leakage computation, which is applied equally to all nodes regardless of whether they are virtual or real. Due to this, we did not observe any significant difference in the RTT distributions obtained when virtual and physical nodes are used.

Protection against alias resolution. Several alias resolution techniques have been proposed to discover whether multiple IP addresses belong to the same router, i.e., if such IP addresses are used in the various router's interfaces. Such techniques could allow adversaries to associate a set of virtual IP addresses to their real node, thus breaking the obfuscation logic behind EqualNet. To avoid such attacks, EqualNet offers strong protection against three widely used alias resolution methods [32] known as (i) common source analysis, (ii) common ID analysis, and (iii) common neighbor analysis.

The first method requires to send TCP/UDP probes to routers' interfaces with unused ports which trigger routers to send *ICMP_Port_Unreachable* responses (e.g., mercator [23], iffinder [4]). With this method, one can conclude that any set of tracing responses containing a common source address originate from the same router. To defend against adversaries who employ this method, EqualNet modifies the tracing response source IP address so that it is the same as the tracing request destination IP address. The second method leverages the way IP IDs are generated in the tracing responses to discover if two tracing responses come from the same router (e.g., ally [40], MIDAR [33], radargun [16]). To defend against this method, EqualNet randomizes the IP ID field within tracing responses. The third method finds the common neighbor by inspecting the IP addresses within the obtained tracing responses. Essentially, if two different IP addresses observed at the same level of TTL hops are adjacent with the common IP address reported in the next hop, those two addresses are likely to be from the same router (e.g., APAR [24], Kapar [32]). To defend against this, EqualNet ensures that no two real or virtual nodes are connected to the same logical node.

To verify that EqualNet can effectively prevent adversaries from using any of these aliasing methods, we conducted a series of experiments with Scamper [9], Iffinder [4], kapar [5], three open-source and widely-used alias resolution tools that support the aforementioned techniques. We generated 1000 tracing-type packets in AS 35132 with the obfuscation threshold τ_I=60% and obtained 449 IP addresses as the result of

[12]This includes the case where EqualNet replies with a real IP address and the one where EqualNet replies with a fake IP address.

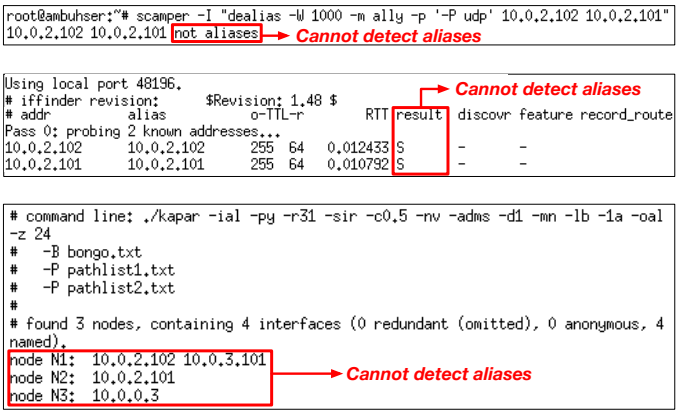


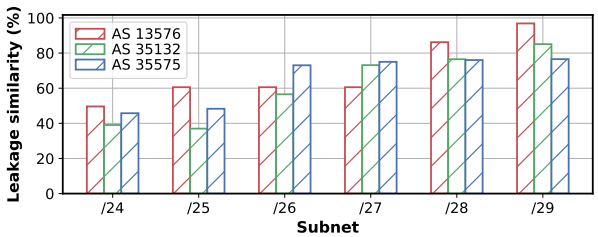
Fig. 17. The results of alias resolution tested with Scamper (top), Iffinder (middle), and Kapar (bottom).

obfuscation. Then we used the tools to check whether they can identify any real router from the real and virtual IP addresses the adversary would get. Based on the obtained results, we can conclude that adversaries who use such tools would be unable to detect aliases among them due the security mechanisms we developed within EqualNet, as shown in Figure 17.

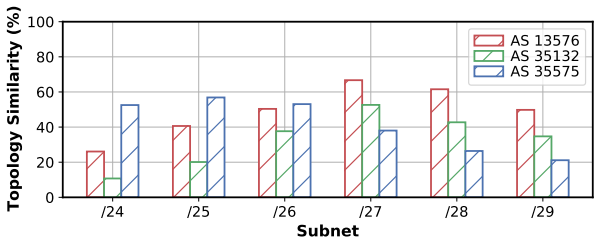
Resistance to topology inference attacks. We also analyze the scenario where adversaries attempt to discover information about the real network topology by trying all possible masks (e.g., from /24 to /29) and aggregating all IP addresses whose mask is identical into a single node (as it is impossible for adversaries to know the number of real nodes within a given subnet)[13]. This process leads the adversary to infer a set of subnet-level topologies (one for each mask). We measured the leakage and the topology similarity between each of the subnet-level topologies inferred by the adversary and the real network topology. As shown in Figure 18a, the adversary can in some cases (e.g., with the /29 prefix) infer the leakage at the subnet-level quite accurately. However, this is not problematic because they cannot know how many nodes are in each subnet, and hence cannot infer the popularity of nodes or links in the network. This is confirmed by the results shown in Figure 18a, which shows that in the worst case the topology produced by the adversary and the real one are similar only by 60%.

Resistance to cross virtual topology attacks. To generate virtual topologies that remain secure long-term, EqualNet not only analyzes and protects a given instance of virtual topology, but also monitors and prevents topology leakage arising from exposing multiple virtual topologies at different times. For the latter, EqualNet employs a set of techniques which guarantee that adversaries who can observe multiple, slightly different virtual topologies gain no knowledge about the network topology and its most popular nodes and links. First, EqualNet adds a fixed number of virtual nodes to each real node which always appear in the virtual topologies generated by EqualNet and hence prevents adversaries from discovering which nodes are the real ones. Second, EqualNet keeps the same network path for a given tracing flow. Thus, an adversary always gets the same information from sending tracing requests through the network path belonging to a specific flow. Finally, whenever a node is starting to look popular, EqualNet creates a new virtual node not only in the node itself but also in all its neighbor nodes. This is important because without adding such neighbor virtual nodes, adversaries could use alias resolution techniques to decrease the security of EqualNet.

[13]Note that here we assume that all nodes have a mask of the same size since this is a best case for adversaries (in that case the number of masks to try is relatively small). However, in practice, with the emergence of CIDR routers are likely to use prefixes of many different sizes.



(a) Leakage similarity for various prefixes (the *lower* the leakage similarity, the more secure the virtual topology is).



(b) Topology similarity for various prefixes (the *lower* the topology similarity, the more secure the virtual topology is).

Fig. 18. Leakage and topology similarity between real network topologies and the subnet-level topologies inferred by the adversary.

Resistance against Denial-of-Service attacks. Over the last decade, several Denial-of-Service (DoS) attacks have been proposed whose goal is to crash or waste the resources of the SDN controller [44], [39]. In this setting, adversaries could attempt to execute such attacks by injecting large amounts of tracing packets to the network. One possible way to mitigate these attacks would be to use a distributed SDN controller [17], [34]. Note that network operators typically rely on a cluster of SDN controllers (i.e., one main controller and various replicas of it) to realize the control plane. This way, if the main SDN controller fails or crashes, any of the replicas can immediately take over, leading to more robust and secure SDN networks. Another possible DoS attack against our solution could be carried out by adversaries who aim to exhaust all available IP addresses. However, as adversaries cannot know the popularity of nodes and links, they cannot influence EqualNet to create a large number of virtual nodes.

VIII. Conclusion

Protecting networks from Link Flooding Attacks (LFAs) is challenging. Past work has addressed some parts of this problem, but limitations of security and usefulness still remain. We have introduced a practical and secure network topology obfuscation solution which prevents adversaries from targeting bottleneck nodes and links by equalizing tracing flow densities. Our solution retains the utility of path tracing information to ensure that network operators can still use path tracing tools to debug their networks.

ACKNOWLEDGMENT

We would like to thank the reviewers for their constructive and valuable comments that helped us to considerably improve our paper. We also want to thank Jaehan Kim and Allarna Janson for their help, comments and support. The research leading to these results received partial funding from the EU H2020 Research and Innovation programme under grant agreements No 871793 (Accordion) and No 101016509 (Charity).

REFERENCES

[1] "Can a DDoS Break the Internet? Sure! Just Not All of It," https://arstechnica.com/information-technology/2013/04/can-a-ddos-break-the-internet-sure-just-not-all-of-it/, 2013.

[2] "Large DDoS Attacks Cause Outages at Twitter, Spotify, and other sites," https://techcrunch.com/2016/10/21/many-sites-including-twitter-and-spotify-suffering-outage/, 2016.

[3] "February 28th Github DDoS Incident Report," https://github.blog/2018-03-01-ddos-incident-report/, 2018.

[4] "Iffinder," https://www.caida.org/catalog/software/iffinder/, 2020.

[5] "Kapar," https://www.caida.org/catalog/software/kapar/, 2020.

[6] "North american network operators group (nanog) mailing list," 2020, https://archive.nanog.org/list/archives.

[7] "Open vSwitch," http://openvswitch.org, 2020.

[8] "Ryu," https://ryu-sdn.org/, 2020.

[9] "Scamper," https://www.caida.org/tools/measurement/scamper/, 2020.

[10] "The CAIDA Internet Topology Data Kit (ITDK) - 2020 Jan." https://www.caida.org/catalog/datasets/internet-topology-data-kit, 2020.

[11] "The Traceroute Command in MPLS," https://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/26585-mpls-traceroute.html, 2020.

[12] "Traceroute for linux," http://man7.org/linux/man-pages/man8/traceroute.8.html, 2020.

[13] "Unmetered Mitigation: DDoS Protection Without Limits," https://blog.cloudflare.com/unmetered-mitigation/, 2020.

[14] M. Antonakakis *et al.*, "Understanding the Mirai Botnet," in *Proceedings of the USENIX Security Symposium*. USENIX, 2017.

[15] A. Aydeger, N. Saputro, and K. Akkaya, "Utilizing NFV for Effective Moving Target Defense Against Link Flooding Reconnaissance Attacks," in *Proceedings of IEEE Military Communications Conference*. IEEE, 2018.

[16] A. Bender, R. Sherwood, and N. Spring, "Fixing Ally's Growing Pains with Velocity Modeling," in *Proceedings of the ACM SIGCOMM conference on Internet measurement*. ACM, 2008.

[17] P. Berde *et al.*, "ONOS: Towards an Open, Distributed SDN OS," in *Proceedings of the Workshop on Hot Topics in Software Defined Networking*. ACM, 2014.

[18] J. Cao, Q. Li, R. Xie, K. Sun, G. Gu, M. Xu, and Y. Yang, "The CrossPath Attack: Disrupting the SDN Control Channel via Shared Links," in *Proceedings of the USENIX Security Symposium*. USENIX, 2019.

[19] A. Dhamdhere, D. D. Clark, A. Gamero-Garrido, M. Luckie, R. K. Mok, G. Akiwate, K. Gogia, V. Bajpai, A. C. Snoeren, and K. Claffy, "Inferring persistent interdomain congestion," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2018.

[20] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and Elastic DDoS Defense," in *Proceedings of the USENIX Security Symposium*. USENIX, 2015.

[21] L. C. Freeman, "A Set of Measures of Centrality based on Betweenness," *Sociometry*, 1977.

[22] X. Gao, B. Xiao, D. Tao, and X. Li, "A Survey of Graph Edit Distance," *Pattern Analysis and applications*, 2010.

[23] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map Discovery," in *Proceedings of the IEEE Conference on Computer Communications*. IEEE, 2000.

[24] M. H. Gunes and K. Sarac, "Analytical IP Alias Resolution," in *Proceedings of the IEEE International Conference on Communications*. IEEE, 2006.

[25] ——, "Inferring Subnets in Router-level Topology Collection Studies," in *Proceedings of the ACM SIGCOMM conference on Internet measurement*. ACM, 2007.

[26] Y. Huang, M. Rabinovich, and R. Al-Dalky, "FlashRoute: Efficient Traceroute on a Massive Scale," in *Proceedings of the ACM Internet Measurement Conference*. ACM, 2020.

[27] Y. Jin, S. Renganathan, G. Ananthanarayanan, J. Jiang, V. N. Padmanabhan, M. Schroder, M. Calder, and A. Krishnamurthy, "Zooming in on Wide-area Latencies to a Global Cloud Provider," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019.

[28] Y. Jin, C. Scott, A. Dhamdhere, V. Giotsas, A. Krishnamurthy, and S. Shenker, "Stable and Practical AS Relationship Inference with ProbLink," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. ACM, 2019.

[29] M. S. Kang, D. V. Gligor, and V. Sekar, "SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks," in *Proceedings of the Network and Distributed System Security Symposium*. Internet Society, 2016.

[30] M. S. Kang and V. D. Gligor, "Routing Bottlenecks in the Internet: Causes, Exploits, and Countermeasures," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014.

[31] M. S. Kang, S. B. Lee, and V. Gligor, "The Crossfire Attack," in *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2013.

[32] K. Keys, "Internet-Scale IP Alias Resolution Techniques," *ACM SIGCOMM Computer Communication Review*, 2010.

[33] K. Keys, Y. Hyun, M. Luckie, and K. Claffy, "Internet-scale IPv4 Alias Resolution with MIDAR," *IEEE/ACM Transactions on Networking*, 2012.

[34] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A Distributed Control Platform for Large-Scale Production Networks," in *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*. USENIX, 2010.

[35] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010.

[36] S. B. Lee, M. S. Kang, and V. D. Gligor, "CoDef: Collaborative Defense Against Large-Scale Link-Flooding Attacks," in *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 2013.

[37] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "NetHide: Secure and Practical Network Topology Obfuscation," in *Proceedings of the USENIX Security Symposium*. USENIX, 2018.

[38] S. Ramanathan, J. Mirkovic, M. Yu, and Y. Zhang, "SENSS Against Volumetric DDoS Attacks," in *Proceedings of the Annual Computer Security Applications Conference*, 2018.

[39] S. Shin and G. Gu, "Attacking Software-Defined Networks: A First Feasibility Study," in *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2013.

[40] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Transactions on Networking*, 2004.

[41] R. A. Steenbergen, "A Practical Guide to (Correctly) Troubleshooting with Traceroute," *North American Network Operators Group*, 2009.

[42] A. Studer and A. Perrig, "The Coremelt Attack," in *Proceedings of the European Conference on Research in Computer Security*. Springer, 2009.

[43] S. T. Trassare, R. Beverly, and D. Alderson, "A technique for Network Topology Deception," in *Proceedings of IEEE Military Communications Conference*. IEEE, 2013.

[44] H. Wang, L. Xu, and G. Gu, "FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE/IFIP, 2015.

[45] L. Wang, Q. Li, Y. Jiang, X. Jia, and J. Wu, "Woodpecker: Detecting and Mitigating Link-Flooding Attacks via SDN," *Computer Networks*, 2018.

[46] Q. Wang, F. Xiao, M. Zhou, Z. Wang, Q. Li, and Z. Li, "LinkBait: Active Link Obfuscation to Thwart Link-Flooding Attacks," *arXiv Preprint*, 2017.

[47] L. Xue, X. Ma, X. Luo, E. W. Chan, T. T. Miu, and G. Gu, "LinkScope: Toward Detecting Target Link Flooding Attacks," *IEEE Transactions on Information Forensics and Security*, 2018.

APPENDIX A
OFFLINE OBFUSCATION ALGORITHM

In this section, we detail the steps followed by EqualNet to estimate the number of virtual nodes needed and decide where they need to be created (see Algorithm 2).

Initially, it takes as input the logical topology $\mathcal{G}$, the leakage reduction τ_I (e.g., 40%), and the number of fixed virtual nodes per real node $virtual^{fixed}$ (e.g., 4) and then computes the maximum leakage allowed $leak^{allowed}$ based on the current leakage $leak^{current}$ and the specified leakage reduction (lines 1 to 2). Next, it proceeds to adding a fixed number ($virtual^{fixed}$) of virtual nodes to all real nodes, and to connecting those nodes with their neighbors, forming virtual links (lines 3 to 5).

The algorithm is executed for as long as the current topology leakage $leak^{current}$ is above than the allowed leakage $leak^{allowed}$ (line 6). To reduce $leak^{current}$, the algorithm first chooses the node n_{curr} whose flow density is the highest among all nodes (line 7), retrieves its existing virtual nodes $\mathcal{N}'_{curr}$ and checks whether the flow density of (at least) one of them is below than the allowed leakage $leak^{allowed}$ (lines 8 to 11). If so, the algorithm chooses such virtual node (denoted as n'_{curr}) to lower the current leakage $leak^{current}$ (lines 12 to 13). Otherwise, it does so by creating a new virtual node n'_{curr} (upon checking that there are IP addresses available in the corresponding subnet) (lines 14 to 18). Either with the existing or the newly-generated virtual node, the algorithm aims to equalize the flow density of the current node. For this purpose, it creates virtual nodes in all its previous neighbor nodes (lines 19 to 20). This task is achieved through the sub-procedure EQUALIZEFLOWSFORPREVNODES (line 23). To that end, it first explores all previous nodes for a given current node n_{curr} (line 24). For each previous node n_{prev}, it attempts to find a previous virtual node n'_{prev} that is connected with the current virtual node n'_{curr} (lines 24 to 25). If no previous virtual node is found, it creates a new virtual node for the previous node (as long as there are IP addresses in the previous node) (lines 26 to 30). Following this, it creates an incoming virtual link and splits the flow density to the virtual link l'_{in} and real incoming link l_{in} equally (lines 31 to 33). This process is repeated in a recursive way until virtual nodes are created for all previous nodes (line 34), and then also in the opposite direction for all its subsequent nodes (lines 35 to 47). Whenever a single loop is finished (lines 6 to 20), the algorithm recomputes the current leakage $leak^{current}$ (line 21) and repeats this process again until the leakage is reduced to the desired leakage $leak^{desired}$.

Algorithm 2 Offline virtual node estimation

Require:
Logical topology $\mathcal{G} = (\mathcal{N}, \mathcal{L}, fd^N, fd^L)$,
Leakage reduction (%) $\tau_I \in [0, 100]$,
Number of fixed virtual nodes per real node $virtual^{fixed}$
Ensure: Virtual topology $\mathcal{G}'$

1: $leak^{current} \leftarrow |\max(fd^{\mathcal{N}}) - \min(fd^{\mathcal{N}})|$
2: $leak^{allowed} \leftarrow (100 - \tau_I) \cdot leak^{current}$
3: $\mathcal{N}'_{fixed} \leftarrow$ CREATEVIRTUALNODESINALLNODES($virtual^{fixed}$)
4: $\mathcal{L}'_{fixed} \leftarrow$ CREATEVIRTUALLINKS($\mathcal{G}, \mathcal{N}'_{fixed}$)
5: $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}'_{fixed}$, $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{L}'_{fixed}$
6: **while** $leak^{current} > leak^{allowed}$ **do**
7: $n_{curr} \leftarrow \arg\max_{n \in \mathcal{N}}(fd^N(n))$
8: $N'_{curr} \leftarrow$ GETEXISTINGVIRTUALNODES($n_{curr}, \mathcal{N}$)
9: $n'_{curr} \leftarrow \emptyset$
10: **for all** $n' \in N'_{curr}$ **do**
11: **if** $fd^N(n') < leak^{allowed}$ **then**
12: $n'_{curr} \leftarrow n'$
13: **break**
14: **if** $n'_{curr} = \emptyset$ **then**
15: **if** IPADDRESSAVAILABLE(n_{curr}) $= false$ **then**
16: **return** $\mathcal{G}'$
17: $n'_{curr} \leftarrow$ CREATEVIRTUALNODE(n_{curr})
18: $\mathcal{N} \leftarrow \mathcal{N} \cup n'_{curr}$
19: EQUALIZEFLOWSFORPREVNODES($\mathcal{G}, n_{curr}, n'_{curr}$)
20: EQUALIZEFLOWSFORNEXTNODES($\mathcal{G}, n_{curr}, n'_{curr}$)
21: $leak^{current} \leftarrow |\max(fd^{\mathcal{N}}) - \min(fd^{\mathcal{N}})|$
22: $\mathcal{G}' \leftarrow (\mathcal{N}, \mathcal{L}, fd^N, fd^L)$
23: **procedure** EQUALIZEFLOWSFORPREVNODES($\mathcal{G}, n_{curr}, n'_{curr}$)
24: **for** $n_{prev} \in \mathcal{G}.predecessors(n_{curr})$ **do**
25: $n'_{prev} \leftarrow$ GETCONNECTEDVIRTUALNODE(n_{prev}, n'_{curr})
26: **if** $n'_{prev} = \emptyset$ **then**
27: **if** IPADDRESSAVAILABLE(n_{prev}) $= false$ **then**
28: $\mathcal{N} \leftarrow \mathcal{N} / n'_{curr}$
29: **return** $\mathcal{G}'$
30: $n'_{prev} \leftarrow$ CREATEVIRTUALNODE(n_{prev})
31: $l'_{in} \leftarrow (n'_{prev}, n'_{curr}), l_{in} \leftarrow (n_{prev}, n_{curr})$
32: $\mathcal{L} \leftarrow \mathcal{L} \cup l'_{in}$
33: $fd^{\mathcal{L}}(l'_{in}) \leftarrow fd^{\mathcal{L}}(l_{in})/2, fd^{\mathcal{L}}(l_{in}) \leftarrow fd^{\mathcal{L}}(l_{in})/2$
34: EQUALIZEFLOWSFORPREVNODES($\mathcal{G}, n_{prev}, n'_{prev}$)
35: **procedure** EQUALIZEFLOWSFORNEXTNODES($\mathcal{G}, n_{curr}, n'_{curr}$)
36: **for** $n_{next} \in \mathcal{G}.successors(n_{curr})$ **do**
37: $n'_{next} \leftarrow$ GETCONNECTEDVIRTUALNODE(n_{next}, n'_{curr})
38: **if** $n'_{next} = \emptyset$ **then**
39: **if** IPADDRESSAVAILABLE(n_{next}) $= false$ **then**
40: $\mathcal{N} \leftarrow \mathcal{N} / n'_{curr}$
41: **return** $\mathcal{G}'$
42: $n'_{next} \leftarrow$ CREATEVIRTUALNODE(n_{next})
43: $l'_{out} \leftarrow (n'_{curr}, n'_{next}), l_{out} \leftarrow (n_{curr}, n_{next})$
44: $\mathcal{L} \leftarrow \mathcal{L} \cup l'_{out}$
45: $fd^{\mathcal{L}}(l'_{out}) \leftarrow fd^{\mathcal{L}}(l_{out})/2, fd^{\mathcal{L}}(l_{out}) \leftarrow fd^{\mathcal{L}}(l_{out})/2$
46: $fd^{\mathcal{N}}(n'_{next}) \leftarrow fd^{\mathcal{N}}(n_{next})/2$
47: EQUALIZEFLOWSFORNEXTNODES($\mathcal{G}, n_{next}, n'_{next}$)

APPENDIX B
RESISTANCE TO SUBNET INFERENCE ATTACKS

Important to the security of EqualNet is that adversaries have no means of inferring how many nodes each subnet contains. Otherwise, they could get an idea of how popular are the nodes inside a given subnet. One of the reasons why it is hard for adversaries to obtain this information is because network operators no longer rely solely on the three well-known IP classes to allocate IP addresses to their nodes.

	m=20, k=30	Inference range (k-m+1)	
(69.69.207.38)	**01000101 01000101 110**	**01111 001001**	10
(69.69.207.37)	**01000101 01000101 110**	**01111 001001**	01

Fig. 19. An example of two IP addresses whose first 30 bits are identical (i.e., $k = 30$). We assume the shortest prefix m is 20. The dotted box denotes the inference bit range ($k - m + 1$).

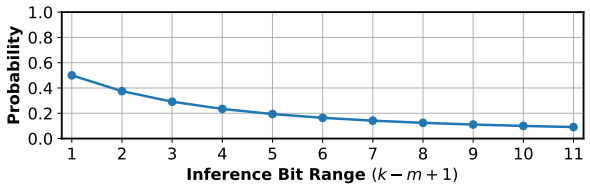


Fig. 20. The probability $P(X_i|E)$ as a function of the inference bit range $k - m$ when $P(A_i)$ follows a uniform distribution, $\frac{1}{k-m}$.

Instead, they commonly use a method known as Classless Inter-Domain Routing (CIDR) which results in a more optimal usage of IP addresses (which are known to be scarce). With CIDR, it is also much more difficult for adversaries to guess which masks are being used.

While adversaries cannot obtain information about the used network masks from the IP addresses in the tracing responses they collect, they can conduct subnet inference attacks to estimate the probability that two IP addresses belong to the same subnet. We model this probability as follows: Let E be an event where two IP addresses have the same first k bits and X_i be an event of having the network mask $i \in \{1, 2, ..., k\}$. Based on the Bayes' theorem, the probability of having the network mask i given the event E is defined by:

$$P(X_i|E) = \frac{P(E|X_i)P(X_i)}{P(E)}, \tag{1}$$

Thus, it is calculated by the likelihood $P(E|X_i)$ and the priors $P(E)$ and $P(X_i)$. We define the likelihood as follows:

$$P(E|X_i) = \frac{1}{2^{k-i}}, \tag{2}$$

as it is the conditional probability of an event of observing the two IP addresses having the k-identical bits given that the mask is i. Thus, Equation (1) is calculated by:

$$\frac{P(E|X_i)P(X_i)}{\sum_j P(E|X_j)P(X_j)} = \frac{\frac{1}{2^{k-i}}P(X_i)}{\sum_j \frac{1}{2^{k-j}}P(X_j)}, \tag{3}$$

where $j \in \{1, 2, ..., k\}$. Now consider a more specific case in which the adversary assumes that the prefix is more likely to be $i > m$ (considering the shortest prefix that a single interface can have within an AS, e.g., /20). Also, let A_i be an event that an adversary chooses a prefix i. Based on the assumption, the probability that the adversary chooses a correct prefix in the range m to k is:

$$\sum_{i=m}^{k} P(A_i)\frac{\frac{1}{2^{k-i}}P(X_i)}{\sum_j \frac{1}{2^{k-j}}P(X_j)} \tag{4}$$

If we assume that $P(X_i)$ and $P(A_i)$ follow a uniform distribution (i.e., all masks in the range are equally likely), the probability indicated by Equation (4) depends on the variable m and k. We define $k-m+1$ as the *inference bit range* which corresponds to the set of possible masks that the adversary can choose (see Figure 19). Note that the longer the range, the lower the probability is of correctly guessing if two IP addresses belong to the same subnet. For example, an inference bit range equal to 10 bits gives adversaries a probability of selecting the mask correctly of 1% probability, as shown in Figure 20. The results obtained may seem counter-intuitive because one would expect that having very similar IP addresses is the worse case for EqualNet. However, as with CIDR any network mask is possible, the best case for EqualNet is when both IP addresses are very similar (since the probability of adversaries discovering the correct mask mask is lower).

We conclude that it is important for network operators to carefully plan not only their IP assignments but also the way IP addresses are allocated to virtual nodes in order to offer strong protection against such attacks. For example, network operators can deliberately assign real nodes similar IP addresses (e.g., 10.0.2.102 and 10.0.2.98) whose subnets are different (e.g., /24 and /29) by choosing the first five bits in the host part in 10.0.2.102 such that they match the corresponding bits in the network part of 10.0.2.98. In such a case, adversaries would most likely believe that these two IP addresses belong to the same subnet when actually they are part of different subnets. Similarly, network operators can assign IP addresses to virtual nodes (e.g., originating from a real node whose prefix is /24, for example, 10.0.2.62/24) such that some of their bits in the host part match one or more existing subnets in the network. To that end, network operators could instruct EqualNet so that it keeps the first four bits in the host part fixed (to simulate a /28 prefix) and chooses any IP address available in the remaining host bits, e.g., 10.0.2.126, and 10.0.2.127. By doing this, the adversary is likely to believe that these two IP addresses belong to the same subnet (with prefix /28) when actually they are virtual nodes of a real node with prefix /24.

APPENDIX C
DISCUSSION

Can LFAs be prevented by blocking tracing packets? One possible mitigation against LFAs would be to disallow path tracing tools like traceroute. However, path tracing tools are fundamental to enable network monitoring and debugging, giving network operators the ability to detect incorrect/sub-optimal routing [41], [26], understand network interconnections [28], [19] or reason about abnormal link latencies [27]. A more reasonable option would be to restrict the usage of such tools so that only other (external) network operators can use them. This could be achieved through the use of specially-configured firewalls used for preventing tracing packets from being sent to network devices. Yet, this approach has several limitations regarding how to verify the legitimacy of incoming traceroute packets, which hampers its adoption. On the one hand, if a simple, non-cryptographic access control mechanism was used (e.g., one that authorizes tracing packets based on their source IP address), adversaries who send tracing packets containing the network identifiers used by (external) network operators could circumvent the firewall and send tracing packets to the network. Furthermore, as tracing packets can be

realized using packet types (e.g., ICMP, UDP and TCP) and ports, it would not be easy for network operators to configure firewalls such that they block tracing packets only. On the other hand, if a cryptographic-based access control mechanism was used, one would require to maintain a large and up-to-date list of cryptographic keys of all (external) network operators who are allowed to use such tools in the network. The main problem with this approach is that it would be very difficult to manage, would be prone to attacks and would considerably increase the network's overhead, e.g., network devices (or middleboxes) would have to perform cryptographic operations to verify the legitimacy of every single incoming tracing packet before it is forwarded to the networking devices.

Does EqualNet protect inter-AS links? Currently, EqualNet does not conceal inter-AS links since this is a task that would require close collaboration between network operators [38], [36]. In future work, we will investigate how to extend EqualNet to obfuscate those links as well as how to incentivize distinct ASes to collaborate together for this purpose.

Can DNS information affect the security of EqualNet? So far, the security of EqualNet (and all previous network obfuscation solutions) relies on the fact that ISPs allocate DNS names to IP addresses that do *not* leak any information about the routers. However, in practice ISPs typically assign DNS names following standard conventions, as described by Spring et al. [40]. While naming conventions are subject to the whims of ISPs and DNS databases, they are not always kept up to date (which mitigates this issue to some extent). In future work we will explore possible ways of assigning DNS names without revealing any information to adversaries.