ConfMask: Enabling Privacy-Preserving Configuration Sharing via Anonymization

Yuejie Wang
Peking University
New York University Shanghai

Qiutong Men
New York University Shanghai

Yao Xiao
New York University Shanghai

Yongting Chen
New York University Shanghai

Guyue Liu
Peking University

Abstract

Real-world network configurations play a critical role in network management and research tasks. While valuable, data holders often hesitate to share them due to business and privacy concerns. Existing methods are deficient in concealing the *implicit* information that can be inferred from configurations, such as topology and routing paths. To address this, we present ConfMask, a novel framework designed to systematically anonymize network topology and routing paths in configurations. Our approach tackles key privacy, utility, and scalability challenges, which arise from the strong dependency between different datasets and complex routing protocols. Our anonymization algorithm is scalable to large networks and effectively mitigates de-anonymization risk. Moreover, it maintains essential network properties such as reachability, waypointing and multi-path consistency, making it suitable for a wide range of downstream tasks. Compared to existing dataplane anonymization algorithm (i.e., NetHide), ConfMask reduces ~75% specification differences between the original and the anonymized networks.

CCS Concepts

• **Networks** → **Network privacy and anonymity**; *Topology analysis and generation*; *Network management*; Data center networks; Routing protocols; Network simulations; • **Security and privacy** → *Security protocols*; *Privacy-preserving protocols*; *Pseudonymity, anonymity and untraceability*.

Keywords

Configuration Sharing, Topology Anonymity, Route Anonymity

ACM Reference Format:
Yuejie Wang, Qiutong Men, Yao Xiao, Yongting Chen, and Guyue Liu. 2024. ConfMask: Enabling Privacy-Preserving Configuration Sharing via Anonymization. In *ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24), August 4–8, 2024, Sydney, NSW, Australia.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3651890.3672217

ACM SIGCOMM '24, August 4–8, 2024, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0614-1/24/08
https://doi.org/10.1145/3651890.3672217

1 Introduction

Real-world network configurations are crucial for many network management tasks and network research. For example, sharing network configurations could enable troubleshooting networks collaboratively [38], optimizing network debugging and verification algorithms [5, 16], and evaluating machine learning models [13]. Unfortunately, few configurations are publicly available and data holders hesitate to share their configurations due to business and privacy concerns.

Prior work [21, 31, 39, 43] use anonymization techniques to hide sensitive information in configuration files before releasing them to third-party or the public. For example, NetConan [21] has identified sensitive attributes that can be *directly* retrieved from configurations, such as IP addresses, AS numbers, and passwords, and used different techniques (e.g., prefix-preserving, substitution) to anonymize them. Unfortunately, it fails to consider two critical types of sensitive information that can be *indirectly* inferred from configuration files, including *network topology* and *routing paths*.

Network topology can disclose valuable business information such as organization structure and scale. It can be reconstructed by parsing the network interface information present in router and host configurations. Similarly, routing paths could expose communication patterns [22, 23]. With router configurations, routing paths can be extracted via simulation or emulation tools (e.g., GNS3 [34], Batfish [16], and Minesweeper [5]).

In this work, we focus on anonymizing topology and routing paths in configurations, to reduce concerns about configuration owners publishing the data. Different from prior work (e.g., [19, 22, 23, 25, 26, 28, 30, 31, 36, 39, 45]), anonymizing these datasets poses several key challenges:

- **Multi-source de-anonymization:** Prior efforts such as graph anonymization [19, 25, 26, 45] and trace anonymization [28, 36] focus on individual data fields such as graphs and IP addresses. However, topology and routing paths are tightly coupled and any modification made to one will inevitably impact the other. This makes them vulnerable to multi-source de-anonymization and requires more sophisticated anonymization techniques.
- **Routing complexity:** Anonymizing routing paths is very challenging due to the complexity of forwarding rules construction, which could involve different routing protocols across multiple nodes [16]. Prior work [22, 30] anonymizes routing behavior by generating random data plane paths. While this approach could obfuscate original routing paths, it may also result in dramatically different network behavior, limiting its use cases (e.g., protocol debugging).

- **Scalability-privacy tradeoff:** It is difficult to effectively anonymize large networks. Complex anonymizing techniques might take hours to anonymize even moderate-size networks, while simple techniques yield poor privacy against re-identification attacks. Striking a balance between privacy and scalability requires not only algorithm optimizations but also theoretical guidelines.

The design of ConfMask tackles these challenges systematically, which involves thoroughly exploring the trade-offs of different anonymization techniques and rigorous theoretical analysis. Specifically, we make three core contributions:

- **Configuration utility formulation:** We propose *functional equivalence* that defines conditions to ensure the usability of anonymized network configurations for a wide range of tasks. To design an efficient anonymization algorithm, we further define *strong functional equivalence* conditions which are easy to implement and provide sufficient conditions to imply functional equivalence (with a formal proof). They are defined for IGP protocols (link-state-based e.g., OSPF and distance-vector-based e.g., RIP) and the BGP protocol.
- **Consistent configuration anonymization workflow:** We propose a workflow that splits the configuration anonymization process into separate stages: topology, route, and PII anonymization add-ons. By incrementally anonymizing different types of data, our proposed workflow ensures that the anonymized configurations can effectively defend against cross-data de-anonymization. The workflow can also adapt different anonymization algorithms per-stage.
- **Efficient route anonymization algorithm**: Our route anonymization algorithm achieves high privacy and can scale to large networks. Our key idea is to decompose complex global route analysis into fast local table lookups that can be performed on each router. For the utility goals, our algorithm converges quickly, and for the anonymity goals, we use a randomized approach to reduce the risk of de-anonymization.

Our implementation of an end-to-end system ConfMask is, to the best of our knowledge, the first prototype designed to anonymize network topology and routing paths for the purpose of configuration sharing. We evaluate ConfMask using eight networks, covering different network scales, structures, and protocols. Our results show that (1) ConfMask can preserve all host-to-host routing paths exactly, while ensuring k-degree anonymity and route anonymity, (2) ConfMask can anonymize large networks (with hundreds of routers) in ∼6 minutes, small networks in seconds, injecting ∼10% lines of configuration files.

Ethics: This work does not raise any ethical issues.

2 Background

In this section, we first highlight the necessity of sharing network configurations with third-party clients in §2.1. Then we discuss critical sensitive configuration information and the limitations of existing anonymization approaches in §2.2.

2.1 Motivating Scenarios

Collaborative debugging: Small to medium-sized enterprises often rely on community forums such as StackExchange and mailing lists for network operations due to limited budgets. They usually post only partial configurations online [2–4, 17] to protect the private information of their companies. However, because of insufficient information provided on network configurations, the resolution and response time of such forums are often poor. A recent survey [38] found that it takes on average a week for collaborators to understand a question, and 81% of the questions require multiple iterations to fill the information gap. Disclosing the full configurations of a network would make collaborative debugging much more efficient. Besides efficiency issues, attacks such as topology inference can be launched even with only partial configurations, revealing confidential information [20, 44]. Larger language models like ChatGPT have proven useful for network troubleshooting [13]. However, any information included in the chat transcript, such as sensitive network configurations, can be disclosed to many people (e.g., affiliates, vendors, service providers) according to OpenAI [35]. Therefore, collaborative debugging demands a privacy-preserving approach to share the full configurations of a network.

Sharing configuration for research purposes: Real-world network configurations are crucial for network research, such as network verification [5, 16] and evaluating machine learning models [13]. However, only a limited number of real network configurations are made public (e.g., by Internet2 [1]) for research purposes due to concerns about confidential (commercial) and private (personal) information. While TopologyZoo [24] contains hundreds of topologies from internet measurements, it does not include realistic network configurations that can be studied. NetComplete [15] provides a synthetic way to generate configuration files, but its predefined rules limit the flexibility in routing utility specifications.

2.2 Threat Model and Sensitive Information

Although network configurations are valuable for various purposes, data holders are often reluctant to share the original configurations due to potential violations of privacy, company policy, and legal regulations in security threats.

Threat model: Based on the motivating scenarios above, suppose that network owner A shares some configurations with a third-party B, for either troubleshooting or research purposes. we consider a possible adversary B to be able to: (1) read configuration files of the shared network, (2) gain partial knowledge of problematic or expected behavior of the original network in the debugging scenario, (3) access any existing network analysis tools, e.g., simulation and verification. However, we do *not* expect the adversary to have direct access to the physical network and perform actions such as `ping` or `traceroute`, since the owner of the network is supposed to be anonymous throughout the sharing process. Our main target is to protect the privacy of the network owner A, therefore, defending against network attacks is not a typical use-case of our system.

Scope of sensitive information: While configuration files contain a wealth of information, our focus in this paper is on the topology and routing paths of networks.

- **Network topology:** Network topology includes the network design details and may reveal confidential business information such as enterprise scale, organizational structure, and growth rate. It can be inferred by analyzing the interface settings across all configurations. Routers and hosts are represented by nodes in the topology graph, and edges are added by identifying interface pairs that share the same prefix.

- **Routing paths:** The data plane of a network determines how packets are forwarded from one node to another, and it is a sensitive aspect since it may reveal traffic patterns, relations between different parts of an organization, etc. The data plane can be extracted from the configuration files via simulation tools such as Batfish [16], or by utilizing a virtual environment to run the configuration [34].

Non-goals: At this stage, we do *not* consider the number of routers as a private attribute. The number of routers does not necessarily link to the identity or privacy of its owner, for example, numerous small to medium-sized enterprises contain about tens of routers. As mentioned above, the traffic patterns and link structure configured in these routers are the identifiable attributes to conceal, to avoid being cross-referenced and leaking critical information about the enterprise. Without further knowledge of the type of each router, having only the number of routers in a company is insufficient to estimate the network scale or the number of users. For instance, the size of the FIB may vary from 4k to 4M on different routers. Still, we note that the workflow we present later in the paper can be extended to adapt anonymization algorithms that alter the number of routers [41], which may be useful for large-scale networks. We will leave them for future works and discuss them in §9. Also, common vulnerabilities and exposures (CVE) like router vendors and operating systems are beyond the scope of this work, and we suggest maintaining them separately before and after anonymization.

2.3 Limitations of Existing Approaches

Graph anonymization: Existing graph anonymization algorithms [8, 9, 11, 19, 25, 26, 45, 46] are potentially capable of anonymizing network topology, but they are primarily designed for social network scenarios and lack protection for information in computer network contexts, such as IP addresses and routing behaviors.

Data plane anonymization: Prior data plane anonymization techniques either modify the packet forwarding behavior or provide a modified topology with a programmable data plane [22, 30, 42] to obfuscate the routing paths. Nevertheless, they may alter the network's functionality, making the anonymized configurations useless even for legitimate purposes such as troubleshooting.

Configuration anonymization: Existing approaches to configuration anonymization focus on protecting **p**ersonal **i**dentifiable **i**nformation (PII), particularly IP addresses [21, 31, 39, 43], but they fail to protect network topology and routing paths.

Motivation Case Study: Lastly, we provide the case study of a concrete trouble-shooting example. Ideally, the misconfiguration problem should remain solvable after anonymization, but the state-of-the-art data plane anonymization approach (i.e., NetHide [30]) fails to achieve the desired utility.

Figure 1 illustrates a sub-topology of a FatTree-04 network (Net H in Table 2) containing the root cause of the following issue caused by a misconfiguration: users are experiencing high network delay and high packet loss rate from h_A to h_B. To solve the performance degrading issue, the network configurations (Listing 1 and 2) are anonymized and shared with an experienced engineer. The fake link (e_{3-1}, agg_{1-1}) was added by NetHide for routing anonymity.

The root cause of the issue is that router c_2 is mistakenly configured to mark inbound traffic from management subnet agg_{3-1}

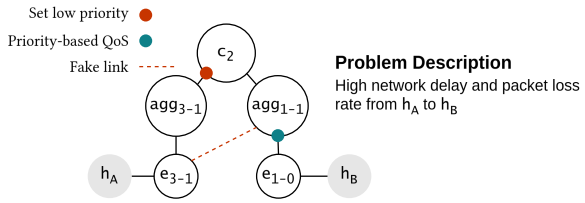


Figure 1: Topology and problem description in the case study

as *low-priority*, which instead should be *high-priority*. Traffic with a priority label (e.g., DSCP) is propagated to agg_{1-1}, where the *low-priority* outbound queue on agg_{1-1} is suffering from congestion. With the NetHide version of the anonymized configuration in troubleshooting, the examined configurations only contain $(e_{3-1}, agg_{1-1}, e_{1-0})$ on the trace path. The visibility of the root cause is blocked by such anonymization, leading the engineer to come up with impractical solutions such as setting the traffic priority on a fake interface. Therefore, a better configuration anonymization solution should preserve the network trace paths between hosts and maintain the `Waypoint` property $(e_{3-1}, agg_{3-1}, c_2, agg_{1-1}, e_{1-0})$ in this case, allowing for precise configuration diagnosis.

```
interface GigabitEthernet1/0/13
  ip address 10.25.17.25/31
  description to-AGG3-1
  traffic-policy mark_agg31_high_priority inbound
!
traffic classifier is_mgmt_traffic
  if-match any
!
traffic behavior remark_mgmt_dscp
  remark dscp af31
!
traffic policy mark_agg31_high_priority
  classifier is_mgmt_traffic behavior remark_mgmt_dscp
!
```

Listing 1: QoS-related configuration of router c_2

```
interface GigabitEthernet1/0/8
  ip address 10.25.7.12/31
  description to-E1-0
  trust dscp
  qos schedule-profile default
!
qos schedule-profile default
  qos wrr 1 to 7
  qos queue 2 wrr weight 10
  qos queue 7 wrr weight 90
!
```

Listing 2: QoS-related configuration of router agg_{1-1}

3 Problem Formulation

Our goal is to anonymize network configurations to safeguard network topology and routing information, and cater to various use cases. We start by defining the problem of network configuration anonymization (§3.1). Then we use several illustrative examples (§3.2) to understand the key challenges (§3.3).

3.1 Terminology and Definitions

Terminology: We introduce the notations to formally represent network configuration $CFG = (G, DP)$, as shown in Table 1. First

Network configuration $\boxed{CFG = (G, DP)}$		**Network anonymization** $\boxed{\widehat{CFG} = \mathcal{A}(CFG) = (\widehat{G}, \widehat{DP})}$	
$r \in R, h \in H$	*routers & hosts*	$\widehat{R} = \cup_{r\in R}\mathcal{A}(r), \widehat{H} = \cup_{h\in H}\mathcal{A}(h)$	*anonymized routers & hosts*
$v \in V = R \cup H$	*network devices*	$\widehat{V} = \widehat{R} \cup \widehat{H}$	*anonymized network devices*
$E = (E_R \subseteq R \times R) \cup (E_H \subseteq R \times H)$	*links*	$\widehat{E} = (\widehat{E_R} \subseteq \widehat{R} \times \widehat{R}) \cup (\widehat{E_H} \subseteq \widehat{R} \times \widehat{H})$	*anonymized links*
$G = (V, E)$	*topology*	$\widehat{G} = (\widehat{V}, \widehat{E})$	*anonymized topology*
$p \in DP$	*data plane*	$\widehat{DP} = \mathcal{A}(DP)$	*anonymized data plane*
Functional equivalence $\boxed{CFG \overset{F}{\simeq} \widehat{CFG}}$	$\exists f$ injective, *s.t.* $\big(f(r) \in \mathcal{A}(r),\ \forall r \in R\big) \wedge \big(f(h) \in \mathcal{A}(h),\ \forall h \in H\big)$;		*topology preservation*
	$\big((h_s, r_1, \cdots, r_n, h_d) \in DP\big) \iff \big((\widehat{h_s}, \widehat{r_1}, \cdots, \widehat{r_n}, \widehat{h_d}) \in \widehat{DP}\big)$, where $\big(\widehat{h_s} = f(h_s),\ \widehat{h_d} = f(h_d),\ \widehat{r_i} = f(r_i),\ \forall i\big)$.		*route equivalence*
Privacy	R is k-anonymized on $\deg_R(r)$;		*k-topology anonymity*
	$\forall p_1 \in DP, \exists p_2, \cdots, p_k \in DP$, s.t. $p_i \sim p_1$ for all $i = 2, \cdots, k$.		*k-route anonymity*

Table 1: Technical cheat sheet

we denote the network topology by $G = (V, E)$, where $V = R \cup H$ are routers (R) and hosts (H) in the network, and $E = (E_R \subseteq R \times R) \cup (E_H \subseteq R \times H)$ are the connections between routers (E_R) and between host-router pairs (E_H). Then we denote the data plane by DP, which is the collection of all host-to-host routing paths in the network. Each routing path $p = (h_s, r_1, \cdots, r_n, h_d)$ is a sequence of nodes in V, through which packets are forwarded from h_s to h_d.

Our goal is to generate an anonymized version of CFG, denoted as $\widehat{CFG} = (\widehat{G}, \widehat{DP})$. The components are obtained through an abstract anonymization mapping $\mathcal{A} : (G \mapsto \widehat{G}) \times (DP \mapsto \widehat{DP})$. The network topology is anonymized as $\widehat{G} = \mathcal{A}(G) = (\widehat{V} = (\widehat{R} \cup \widehat{H}), \widehat{E} = (\widehat{E_R} \cup \widehat{E_H}))$, where

$$\widehat{R} = \mathcal{A}(R) = \cup_{r\in R}\mathcal{A}(r), \quad \widehat{H} = \mathcal{A}(H) = \cup_{h\in H}\mathcal{A}(h),$$

and links can be added between network devices in the $\widehat{V}$. The anonymized data plane $\widehat{DP} = \mathcal{A}(DP)$ consists of all routing paths in the anonymized network, and specifically each routing path $p = (h_s, r_1, \cdots, r_n, h_d) \in DP$ will be anonymized into the set

$$\mathcal{A}(p) = \{\widehat{p} = (\widehat{h_s}, \widehat{r_1}, \cdots, \widehat{r_n}, \widehat{h_d}) \in \widehat{DP};\ \widehat{h_s} \in \mathcal{A}(h_s),$$
$$\widehat{h_d} \in \mathcal{A}(h_d),\ \widehat{r_i} \in \mathcal{A}(r_i), \forall i \in \{1, \cdots, n\}\} \subseteq \widehat{DP}.$$

Problem: We require privacy on each component of CFG. At a high level, the anonymization mapping $\mathcal{A}$ should effectively conceal the original topology and routing paths, preventing an adversary from identifying them from the anonymized version (Definition 3.1 and 3.2). Meanwhile, $\widehat{CFG}$ should satisfy the usability conditions to preserve functional features of the network (Definition 3.3).

Anonymity: We characterize the privacy of G and DP by identifying key attributes and preventing them from being uniquely identified. Note that privacy definitions against membership disclosure (e.g., differential privacy [29]) are not applicable in this case, as they aim to prevent an adversary from learning whether a specific record is included in the data [10]. Instead, we aim to prevent the disclosure of sensitive attributes, which is why we use k-anonymity [40]. By guaranteeing that each data record is indistinguishable from at least $k-1$ other records, k-anonymity provides necessary privacy protection.

- **Topology anonymity:** The key attributes of the topology G of a network are the degrees of its nodes [25]. For each router $r \in R$, let $\deg_R(r)$ denote the number of connections between r and some other router $r \in R \setminus \{r\}$.

  Definition 3.1. *CFG* satisfies *k-topology anonymity* if the collection of routers R is k-anonymized on $\deg_R(r)$.

- **Route anonymity:** We anonymize the data plane to conceal the routing behaviors of a network, whose key attributes are the host connections with the same ingress and egress routers. Let $p = (h_s, r_1, \cdots, r_n, h_d)$ and $p' = (h'_s, r'_1, \cdots, r'_{n'}, h'_d)$ be two routing paths, and we say $p \sim p'$ *if and only if* $r_1 = r'_1$ and $r_n = r'_{n'}$, i.e., they share the same ingress and egress routers.

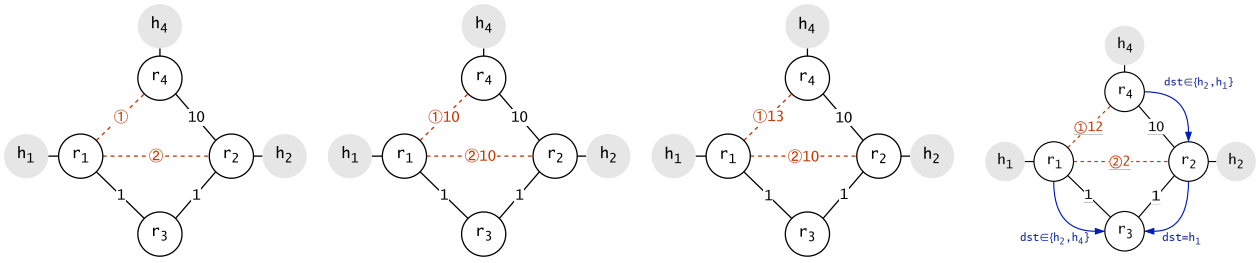
  Definition 3.2. *CFG* satisfies *k-route anonymity* if for each $p_1 \in DP$, there exist at least $k-1$ other routing paths $p_2, \cdots, p_k \in DP$, such that $p_i \sim p_1$ for all $i = 2, \cdots, k$.

Usability: Anonymization techniques (e.g., NetHide [30]) employed to obfuscate original network data may inevitably introduce changes that can potentially impact the utility of the data. To ensure that the resulting $\widehat{CFG}$ is useful for a broad range of tasks, we aim to achieve **functional equivalence** (denoted by $CFG \overset{F}{\simeq} \widehat{CFG}$) which is defined as follows:

Definition 3.3. $\widehat{CFG}$ is *functionally equivalent* to *CFG*, if:

- (Topology preservation) All routers, hosts, and links originally in G remain in $\widehat{G}$.
- (Route equivalence) DP and $\widehat{DP}$ are identical with respect to routing behaviors between hosts in G.

Functional equivalence is useful since it necessitates an *if and only if* condition, which ensures that we do not make over-approximations that can result in falsely identified bugs that do not exist in the original network, nor do we make under-approximations that can cause us to overlook property violations in the original network. Moreover, functional equivalence preserves important routing utility properties, including reachability, path lengths, black holes, multipath consistency, waypointing, and routing loops [6]. The preservation of these properties guarantees the utility of the anonymized



(a) The original topology and the fake edges added (dashed red lines)

(b) Enable OSPF on the fake interfaces with default cost

(c) Set OSPF cost on the fake edges to be sufficiently large

(d) Set OSPF cost to match the lowest cost; the blue arrows represent the PBR policy

Figure 2: The example network

network, which will be further explained in §5.1 and rigorously stated and proved in Appendix B.

3.2 Strawman Approaches and Limitations

Example Network: Consider the enterprise network illustrated in Figure 2a with four routers. The routers r_1, r_2, and r_4 are connected to the computers in their respective departments, represented by h_1, h_2, and h_4, respectively. All inter-router interfaces have OSPF enabled in the same area, and each router advertises the prefix of the respective host. Interfaces connecting $(r1, r3)$ and (r_3, r_2) are configured with OSPF cost 1 instead of the default cost of 10.

The original configurations of the network reveal the topology and routing paths among different departments. For instance, packets from h_1 to h_4 are only reachable through path $(h_1, r_1, r_3, r_2, r_4, h_4)$, which may expose confidential inter-departmental business or geo-relationships and compromise privacy. To anonymize this network, we consider a strawman approach which generates a network configuration with sensitive information hidden but utility preserved.

Step 1. Add fake edges: The obvious way to conceal topology is to add fake links to the network and assign proper IP addresses, as depicted by the dashed edges in Figure 2a. The fake links make it difficult to reconstruct the topology graph by simply excluding the edges with no IP address for each end. However, the fake inter-router interfaces can be easily identified since they do not have OSPF configuration.

Step 2. Configure protocol: Even with the interface IPs added to the routing protocol, there remains the question of fake edge costs, which affect the routing behaviors. Naturally, we have the following three naive options to set OSPF cost.

(i) Use default cost: As shown in Figure 2b, link state information propagates through the new links. Although the topology is harder to reconstruct, the shortest-path-tree (SPT) algorithm used by OSPF will migrate the routing path between h_1 and h_4 from $(h_1, r_1, r_3, r_2, r_4, h_4)$ to (h_1, r_1, r_4, h_4), violating routing-level equivalence above.

(ii) Set a large cost: Another option is to set the cost on all fake edges to a sufficiently large value so that routers always have the least priority on new links (Figure 2c). This approach achieves routing equivalence by exactly preserving routing paths, but no traffic will fly through the fake edges. Applying the SPT calculation precisely identifies these links.

(iii) Match the lowest cost: The third option is to set the cost of new links to match the original shortest path. As shown in Figure 2d, this change can import traffic into fake edges. For example, traffic between h_1 and h_4 is now split between $(h_1, r_1, r_3, r_2, r_4, h_4)$ and (h_1, r_1, r_4, h_4). However, this modification changes the forwarding behavior between (h_1, h_4) from a single path to multiple paths.

Step 3. Fix the original routing: To move traffic from the fake edges back to its original path, we can manually adjust each hop. For instance, we can apply policy-based routing (PBR) to hardcode the next hop on each router for each destination host. As an example, the blue arrows in Figure 2d represent the PBR policy. On r_1, we can match all packets destined for h_2 or h_4 and forward them to r_3, restoring the routing behaviors.

Limitations of the strawman approach: The routing fix above has significant limitations. Firstly, PBR supports only a single next-hop and therefore cannot achieve routing-level equivalence in many scenarios, such as load-balancing paths. Secondly, the configuration lines of PBR leave explicit routing information of the original network. Lastly, a network simulation tool such as Batfish [16] can be used to identify and remove ghost links from the topology, thus revealing the original network topology.

3.3 Challenges

From the example above, we observe that network configuration anonymization poses three unique challenges compared to prior works such as network trace anonymization [28, 32, 36] and graph anonymization [19, 25, 26, 45].

C1. Topology and routing are tightly coupled: They are coupled in the sense that a single modification made to one will usually impact the other. Furthermore, de-anonymization of one data type is possible by referring to the other. As demonstrated in the strawman, the anonymized network topology can be inferred through unconfigured interfaces, shortest path tree, and routing simulation.

C2. Routing complexity makes utility preservation difficult: A small change to a configuration file could affect the routing behavior of multiple components, as demonstrated in Step 2 of the strawman. This results in two issues: (i) some seemingly possible anonymization solutions will negatively impact the utility, as shown in Step 3; (ii) a solution that works for one type of routing protocol (e.g., BGP) may not work for other types of protocols (e.g., OSPF).

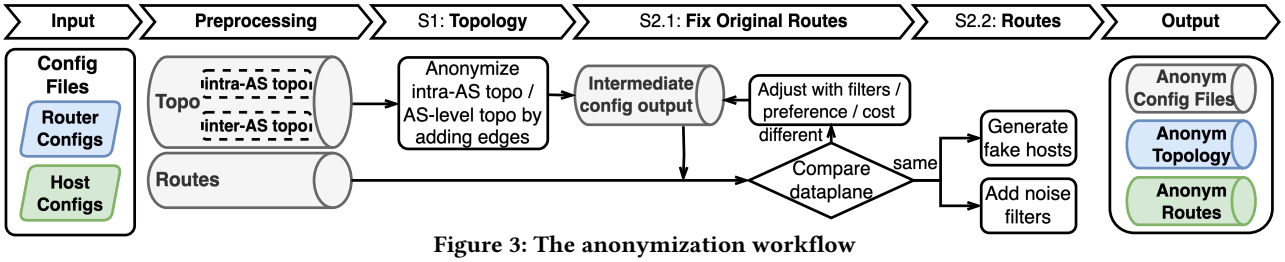


Figure 3: The anonymization workflow

C3. Combined effects make anonymizing large networks even more challenging: The previous two challenges are compounded for large networks. Large networks have more nodes, edges, and routing paths and are typically configured with complex routing mechanisms such as load balancing and aggregation. Consequently, solutions that work for small networks may not scale well to large networks.

4 ConfMask Design

In this section, we present the design of ConfMask, which can effectively anonymize network configurations while achieving functional equivalence, including the overall workflow and each phase of anonymization.

4.1 Anonymization Workflow

Our anonymization workflow design is illustrated in Figure 3, consisting of a preprocessing step, two major anonymization steps (topology anonymization and route anonymization), and other add-on steps of existing anonymization algorithms. The preprocessing step computes the topology and routes of the original network as the baseline. Step 1 (§4.2) modifies the configuration files to generate the anonymized network topology in intra-AS level and inter-AS level. Step 2 (§4.3) then tackles route processing, which is further divided into two parts: Step 2.1 (§5.1) ensures route equivalence and Step 2.2 (§5.3) anonymizes the routes in the new network.

The order in which we anonymize topology and routing behavior is essential. It is safe to modify the routing behavior after fixing the topology but not vice versa. If we modify the topology after the routes have been fixed, then the new links added to the topology would have no impact on the routing behavior, thus not containing any traffic. This would make them suspicious and easy to de-anonymize.

4.2 Topology Anonymization

The topology of a network is regarded as a simple graph that consists of routers R and the links between routers E_R (we exclude hosts during topology anonymization). We adopt an existing graph anonymization algorithm to generate a fake topology and modify the configuration files accordingly. Existing graph anonymization algorithms fall into two types: those that preserve the same set of nodes in the graph and only modify edges [25], and those that add virtual nodes during anonymization [41]. Both types of algorithms automatically fulfills the *topology preservation* requirement in functional equivalence. We choose the first type of algorithm as it results in an anonymized topology that is more similar to the original network, but we note that the second type is applicable to our workflow as well.

For any new edge generated by the graph anonymization algorithm, we add a new link to the network by modifying the configuration files of the corresponding routers (details in §6). There are hardly any constraints for applying this method, except for maybe the hardware capabilities such as the number of interfaces on a router. But the highest node degree remains unchanged in this algorithm, and we have never encountered a case where we are unable to further add interfaces to a router in ConfMask.

BGP is a special case where we need to view the topology in two levels: the routers in each autonomous system (AS) form a simple graph, and on top of that each AS is treated as a (super)node so the network of ASes is treated as a simple graph as well. In particular, we adopt the topology anonymization algorithm for each AS independently, adding edges using the aforementioned strategy until the definition of topology anonymity within each AS is satisfied. After that, we anonymize the network of ASes by viewing two ASes as interconnected as long as one of their border routers is interconnected. A new edge between two ASes is implemented by adding an edge between two randomly chosen border routers in the two ASes respectively.

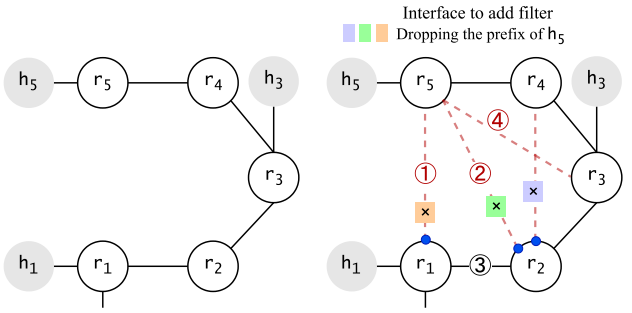
Our approach achieves *topology preservation* by retaining the existing nodes and edges in the new topology. We also ensure that only new configuration lines are added without deleting any existing configurations. However, adding links to the current network topology can affect the routing behavior of the network, potentially violating route equivalence.

Routing impact: There are two types of changes that could affect the original routing paths. The first type involves modifying existing next hops to fake interfaces. For example, new route advertisements can be sent through the fake links, possibly causing the fake interfaces to be selected as the next-hop interface in the routing table. The second type of change involves altering route preferences through original neighbors. For example, the fake links could change the preference order (e.g., OSPF cost, BGP local preference) of the routes learned from real neighbors.

Figure 4a is part of a BGP example network, and Figure 4b shows the corresponding network after anonymizing topology with $k = 3$. Destination h_5 has three candidate paths on r_1: ① is currently selected as the best path and used for forwarding, ③ is the second-best path since the route decision process follows a shorter AS path with higher priority. We will discuss in §4.3 how to fix the incorrect paths (e.g., ①), thereby achieving route equivalence.

4.3 Route Processing

Route processing involves two steps: (1) fixing changed routes to achieve route equivalence, and (2) and anonymizing original routes to achieve k-route anonymity. In this section, we focus on the first



(a) Example network (b) 3-topology anonymity

Path matchs original DP
First different hop from original DP

Test	Trace Path
original	h_1-r_1-r_2-r_3-r_4-r_5-h_5
iter #1	h_1-r_1-r_5-h_5
iter #2	h_1-r_1-r_2-r_5-h_5
iter #3	h_1-r_1-r_2-r_4-r_5-h_5
iter #4	h_1-r_1-r_2-r_3-r_4-r_5-h_5✓

(c) Trace paths for strawman 2

Next-hop interface matchs original DP
Different next-hop interface from original DP

Looked in	r_1	r_2	r_3	r_4	r_5
original	→r_2	→r_3	→r_4	→r_5	→h_5
iter #1	→r_5	→r_5	→r_4	→r_5	→h_5
iter #2	→r_2	→r_4	→r_4	→r_5	→h_5
iter #3✓	→r_2	→r_3	→r_4	→r_5	→h_5

(d) Local FIB lookup solution

Figure 4: Example network anonymization and fixing its fix paths to achieve route equivalence

step by elucidating the challenges of correctly filtering wrong routes and presenting our solution.

Strawman 1: Simply dropping all incoming host prefixes on every fake interface can correctly fix changed routes. However, it is vulnerable to de-anonymization due to the unified pattern on each router. Listing 3 is a configuration example for Cisco IOS that filters out every host on r_2 using the `distribute-list` filtering feature. The lines 2–5 are newly added on each router for each fake neighbor. An adversary can potentially identify the fake interfaces that always bind to a minimal subset of dropped prefixes shared by all routers.

```
1 router bgp 20
2   neighbor {r5} distribute-list RejPfxs in Et0/2
3   neighbor {r4} distribute-list RejPfxs in Et0/3
4 ip prefix-list RejPfxs seq 5 deny {h1-prefix}
5 ip prefix-list RejPfxs seq 10 deny {h4-prefix}
6 ip prefix-list RejPfxs seq 15 deny {h5-prefix}
```

Listing 3: Distribute-list configuration example

Strawman 2: Consider filtering only the host prefixes for which traffic passes through on each fake interface. This breaks the unified pattern of the approach above. To do this, we can use a simulation tool to extract the data plane from the configuration files. For each host connection $h_a \rightarrow h_b$, we run `traceroute`(h_a, h_b) before and after topology anonymization and identify the first different hop r_i that is closest to h_b (i.e., $r_{i+1}, \cdots, r_k$ are part of an original path from h_a to h_b but r_i is not). We add a route filter to the interface $int_{i,i+1}$ connecting r_i and r_{i+1} to filter out routes whose destination IP contains the host h_b.

For instance in Figure 4b, `traceroute`(h_1, h_5) would result in the routing path (h_1, r_1, r_5, h_5), and we compare it with the original path $(h_1, r_1, r_2, r_3, r_4, r_5, h_5)$. r_1 is the first different hop closest to h_5, and we fix this by filtering out the prefix of h_5 on $int_{1,5}$. The next-hop for h_5 should then fall to r_2, and we need to check against the new data plane if the new path matches the original ones. However, due to the protocol dynamics nature of BGP, which always selects a local equilibrium rather than a global optimum [18], simply removing the filtered route from the data plane is not sufficient to infer the new data plane. Therefore, we need to use a simulation tool to obtain the modified configuration to extract the new data plane.

We can repeat the re-simulation and filtering process until the resulting data plane is the same as the original network. As shown in Figure 4c, this approach fixes path inconsistencies one hop at a time, and for the $h_1 \rightarrow h_5$, it takes four iterations to correct the path. Although this approach is less vulnerable, re-simulation and FIB lookup jobs complexity makes the approach impractical.

Our approach: Note that the filters added in Strawman 1 are based solely on local information in the configuration files, such as which interfaces are fake. While this approach is simple, it lacks the ability to target some specific routes. Strawman 2, on the other hand, demonstrates that by combining local information with route information, it suffices to add filters only to certain destinations on a fake link. However, in the second strawman approach, the process of `traceroute` is time-consuming. If we decompose a `traceroute` request into a chain of FIB lookups, we observe that there are many unnecessarily duplicated FIB lookups in the second approach due to multiple host pairs sharing the same part of the path. For example, in the first iteration shown in Figure 4c, h_5 has been looked up in the same FIB twice on r_5, once during `traceroute`$(h_1, h_5) = (h_1, r_1, r_2, r_5, h_5)$ and the other during `traceroute`$(h_3, h_5) = (h_3, r_3, r_5, h_5)$.

Based on this observation, our final solution (§5) aims to balance between the two approaches to achieve an acceptable complexity by significantly reducing the number of time-consuming jobs without forming patterns in the configurations that may lead to de-anonymization.

5 Route Equivalence and Anonymization

In this section, we propose strong functional equivalence conditions (§5.1) for specific types of routing protocols and prove that they are sufficient to infer functional equivalence. Following the conditions, we design algorithms to achieve *route equivalence* (§5.2) and k-route anonymity (§5.3). Finally, we analyze the complexity complexity of our algorithms in §5.4.

In §4 we already restrict our solution to only generating fake hosts and fake links without adding any fake routers, so that $\mathcal{A}(R) = R$. As for the hosts, regardless of the fake hosts added, we require that each host $h \in H$ corresponds to a unique real host $\widehat{h} \in \mathcal{A}(h)$, which we denote by $\mathcal{A}^0(h) = \widehat{h}$.

5.1 Strong Functional Equivalence

As our definition of functional equivalence is based on the routes between any two hosts, it is not straightforward to find an efficient solution that guarantees this requirement. Alternatively, we propose a set of conditions that are sufficient to imply functional

equivalence while being easy to evaluate and implement, which we call **strong functional equivalence (SFE) conditions**. By embedding SFE conditions into the algorithm, we can guarantee that the anonymized network we generate automatically fulfills functional equivalence with the original network. The SFE conditions would vary for different types of routing protocols, as we will state next.

Distance-vector protocols: In distance-vector routing protocols such as RIP and EIGRP, each router advertises its distances to its neighbors and receives the advertisements from its neighbors until convergence. In order to guarantee functional equivalence, we thus require each routing path in CFG to be imported in $\widetilde{CFG}$, and each advertisement learned by a router in $\widetilde{CFG}$ to be a valid advertisement in CFG as well. This invokes the following SFE conditions:

(1) If $e = (r, r') \in E_R$, then we require $\widehat{e} = (\mathcal{A}(r), \mathcal{A}(r')) \in \mathcal{A}(E_R)$, with $\widehat{e}$ having the same link properties as e. Similarly, if $e = (r, h) \in E_H$, then we require $\widehat{e} = (\mathcal{A}(r), \mathcal{A}^0(h)) \in \mathcal{A}(E_H)$, with $\widehat{e}$ having the same link properties as e.
(2) If any route is imported from some neighbor $\widetilde{v'} \in \mathcal{A}(V)$ to the routing table of $\widehat{v} \in \mathcal{A}(V)$, there must exist $v \in V$ and $v' \in V$, such that $\widehat{v} = \mathcal{A}(v)$, $\widetilde{v'} = \mathcal{A}(v')$, and v' is some neighbor of v.

The first condition ensures that all the links in the original network still exist as candidates for forming routing paths in the anonymized network, and that the preference for each of these candidates remains unchanged. Yet since we are adding new links, there may be additional paths that are preferred over these candidates. The second condition then ensures that no additional routing paths will be accepted in the anonymized network, so that functional equivalence would hold.

BGP: The SFE conditions for BGP are defined based on those for distance-vector protocols. First, each AS must satisfy the SFE conditions for distance-vector protocols to maintain the routes within each AS. On top of that, the network of ASes must also satisfy the SFE conditions for distance-vector protocols, so that the inter-AS routing also remains unchanged to fulfill functional equivalence.

Link-state protocols: The advertisement of a link-state protocol such as OSPF sends the neighboring relation across the network. Each router learns about the network topology from the advertisement of its neighbors and computes the routing path with the lowest cost. In order to guarantee functional equivalence, we thus require each routing path in CFG to remain of lowest-cost in $\widetilde{CFG}$. This invokes the following SFE conditions:

(1) If $e = (r, r') \in E_R$, then we require $\widehat{e} = (\mathcal{A}(r), \mathcal{A}(r')) \in \mathcal{A}(E_R)$, with $\texttt{cost}(\widehat{e}) = \texttt{cost}(e)$. Similarly, if $e = (r, h) \in E_H$, then we require $\widehat{e} = (\mathcal{A}(r), \mathcal{A}^0(h)) \in \mathcal{A}(E_H)$, with $\texttt{cost}(\widehat{e}) = \texttt{cost}(e)$.
(2) If $\widehat{e} = (\widehat{v}, \widetilde{v'}) \in \mathcal{A}(E)$ but $e = (\mathcal{A}^{-1}(v), \mathcal{A}^{-1}(v')) \notin E$, then we require either $\texttt{cost}(\widehat{e}) > \texttt{min\_cost}(\mathcal{A}^{-1}(v), \mathcal{A}^{-1}(v'))$, or $\texttt{cost}(\widehat{e}) = \texttt{min\_cost}(\mathcal{A}^{-1}(v), \mathcal{A}^{-1}(v'))$ but $\widehat{e}$ is rejected. Here $\texttt{min\_cost}(v, v')$ denotes the minimum link cost between v, v'.

Similar to the conditions for distance-vector protocols, the first condition ensures that all the links in the original network still exist as candidates for forming routing paths in the anonymized network, and that the costs of these candidates remain unchanged. Yet since we are adding new links, these may create new paths with lower costs. The second condition then ensures that shorter paths will not exist, and if equal-cost shortest paths are generated, we reject them so that the data plane will be fixed back to the original.

We claim that the SFE conditions defined above imply functional equivalence (Theorem A.4, Appendix A). Thus by fulfilling the SFE conditions, our algorithm can produce anonymized networks that automatically satisfy functional equivalence.

As the *topology preservation* requirement is already satisfied during topology anonymization in §5.3, we introduce our approach to further satisfy the *route equivalence* requirement by fulfilling the SFE conditions.

5.2 Route Equivalence Algorithm

In this section, we present our algorithm to fulfill the SFE conditions. We satisfy the first condition for both types of routing protocols by ensuring that no existing configuration is modified or deleted, so that any link in the original network remains the same in the modified network. We also fulfill the second condition by only adding new lines to the configuration files. For distance-vector protocols, we add filters to remove routing table entries that violate the second condition. For link-state protocols, we first set the cost of each fake link to the minimum cost between the two routers in the original network, such that $\texttt{cost}((r, r')) = \texttt{min\_cost}(r, r')$. Then, we can satisfy the second condition by adding filters in the same way as distance-vector protocols.

Algorithm 1 Route Equivalence Algorithm

Input: CFG: the original network; $\widetilde{CFG}$: the intermediate network after topology anonymization
Output: Functionally equivalent $\widetilde{CFG}$

```
repeat
    for ⟨r̃, h̃_d, nxt⟩ ∈ D̃P do
        if nxt ∉ DP[r̃, h̃_d] and (r̃, nxt) ∉ E then
            Add filter on r̃ to deny h̃_d from nxt
until SFE conditions are fulfilled
```

As shown in Algorithm 1, we take the intermediate $\widetilde{CFG}$ after topology anonymization, and apply an iterative approach similar to strawman 2 to fix the difference between the original data plane DP and the intermediate data plane $\widetilde{DP}$. Only the routing paths that involve fake links need to be considered in this step, so in each iteration, we look up every host destination on every router and deny the routing paths pointing to some fake neighbor as the next hop. Note that this approach may take multiple iterations to converge, as routers do not have a global view of the network when choosing their next hop. Therefore, adding filters in one iteration does not guarantee that the correct next hop will be selected in the succeeding iteration.

However, compared with strawman 2, our approach can significantly reduce the number of iterations needed for convergence. This is because strawman 2 addresses only the wrong next-hops from a source host to a destination host, leaving some wrong next-hops on partial paths from routers to the destination host unchanged. These next-hops may still be selected in further iterations with strawman 2. On the other hand, our algorithm examines all routing table entries of each router within each iteration, thus fixing as least at many next-hops as strawman 2.

Compared with the strawman approaches, our approach has the following trade-offs: we looked up hosts on all routers, while some of them do not necessarily appear in any path targeting the

specified host. However, this overhead is relatively small since our approach converges quicker and has fewer iterations, resulting in a significantly lowered total time (§7).

5.3 Route Anonymity Algorithm

To achieve our definition of route anonymity in §3.1, we first add $k-1$ copies for every host (i.e. same configuration as the original host except for hostname and IP address) and connect them to the same ingress router. Notice that in §5.1, we fix the data plane difference by adding filters to reject false routing table entries. We use a similar approach to anonymize host connections which involve fake hosts with the same pair of ingress and egress routers, so the adversary cannot infer that the routes influenced by distribute-lists are valid routes from the real network. For the $k-1$ host copies connected to the same egress router, the idea is to add filters at different hops along the route so that the routes from one ingress router to different hosts on the same egress router can achieve an upper-bound of k-anonymity. The challenges here are: assigning the appropriate IP addresses for the fake hosts and choosing where to add filters.

For each fake host, we choose a new IP that is not included by any network that appeared in the original network configurations. This will ensure that the routing behavior regarding the fake hosts is in our control given that all the existing configuration lines are unchanged, and that the route filters we added for the real hosts and the fake hosts will not affect each other. Each fake host is connected to its ingress router by a pair of matching fake interfaces, and its network is added to the protocol running on this router. Then, we use a randomized algorithm (Algorithm 2) to add filters for the fake host destinations while preserving the original reachability. We use $p = 0.1$ for the evaluations that follow.

Algorithm 2 Route Anonymization Algorithm

Input: $\widetilde{CFG}$: the intermediate network after generating $k-1$ fake hosts for each real host; p: noise coefficient
Output: Route anonymized network $\widetilde{CFG}$

1: **for** $\widetilde{r} \in \widetilde{R}$ **do**
2: $\text{DstH}_{old}[\widetilde{r}] \leftarrow$ reachable fake hosts from $\widetilde{r}$ in $\widetilde{CFG}$
3: **for** FIB entry containing fake hosts fh **do**
4: with prob. p, add filter on $\widetilde{r}$ to deny fh from nxt
5: $\text{DstH}_{new}[\widetilde{r}] \leftarrow$ check host reachability of $\widetilde{r}$
6: **for** fake host $fh \in \text{DstH}_{old}[\widetilde{r}] \setminus \text{DstH}_{new}[\widetilde{r}]$ **do**
7: Remove the filter for fh

5.4 Discussion

Time complexity: As we trade `traceroute` requests in the second strawman approach for router local FIB lookups, the remaining most time-consuming job in our workflow is data plane simulation, i.e., the number of iterations in our algorithm. The number of iterations can be strictly upper-bounded by the number of edges added during topology anonymization, because in each iteration, for a specific destination, if there exists some wrong next hop in the data plane, at least one fake link will be added with the filter regarding this destination. But we notice that, in general, we are able to satisfy SFE conditions within a smaller number of iterations.

Modified configuration lines: Our modifications to the configuration files include fake hosts, interfaces, and filters. The filters are added during step 2.1 and step 2.2 in our workflow. In Algorithm 1, at most $O(|H|(|\widetilde{E}| - |E|))$ filters are added to the configurations. In Algorithm 2, the expected number of filters is less than $O(|R| \cdot |H|)$ The number of configuration lines for $(k_H - 1) \cdot |H|$ fake hosts and corresponding interfaces are deterministic. Other fake interfaces are generated during topology anonymization, for which we adopted the existing graph anonymization algorithm, so we will not elaborate on its complexity here.

Privacy analysis: The ConfMask workflow is a general approach to anonymize the network configurations in a consistent manner, considering both the control plane and the data plane to satisfy the desired privacy while preserving **functional equivalence** (§3.1) for utility. We provide formal abstractions of network attributes and the theoretical proof of our privacy guarantees, which ensures k-anonymity in node degrees for the topology and routing paths for the data plane.

Limitations: In the sections above, we demonstrate the key components of the ConfMask workflow applying a basic k-anonymity algorithm for topology and routes, which are known to be susceptible to some deanonymization techniques such as related datasets and membership inference [33]. But we emphasize that ConfMask can be enhanced with other anonymization algorithms providing stronger anonymity guarantees [11, 46] or providing anonymization for the number of routers [41].

6 Implementation

We implement ConfMask in Python with ~4000 lines of code and use pybatfish [16] for the network simulations in our workflow (Figure 3). ConfMask currently supports protocols such as BGP, OSPF, RIP in Cisco routers, and it is easily extendable to more protocols and vendors (limited to Batfish support for now) using the same logic. In the preprocessing step, we record the original topology and routes in the network simulated from the input configuration files. Then, we parse the configuration files to separate configuration lines for interfaces, protocols, filters, and leave the lines that do not fall within these categories unchanged throughout the workflow.

Topology anonymization: Step 1 adds new interface configurations and adds network to existing protocol configurations on a router. We use an existing algorithm [25] to satisfy $k-$anonymity for the network topology, but if a different topology anonymity is used, any existing graph anonymization algorithm that satisfies the limitation mentioned in 4.2 can be easily adopted to topology anonymization. Suppose that a new edge needs to be added between router r_1 and r_2, we generate an IP prefix that is not in the original network, and add interfaces to r_1 and r_2 with matching IPs in the new prefix. Then, we add this network prefix to the existing routing protocol configurations on both routers.

Route anonymization: We implement the filters added in Step 2 using distribute-list configurations on the Cisco routers. Apart from distribute-list configurations, we also need to add lines to existing configurations to apply the filter to a certain interface or neighbor. For example, in OSPF networks, new lines are added to the *next_hop* interface to apply the filter, while in BGP networks, new lines are added to the *bgp neighbor* configuration.

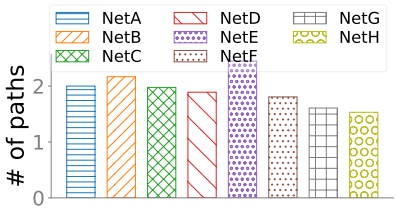


Figure 5: Average number of distinct paths between edge routers in all networks, with $k_R = 6$ and $k_H = 2$

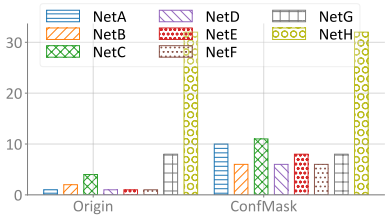


Figure 6: Minimum number of nodes of the same degree in all networks, with $k_R = 6$ and $k_H = 2$

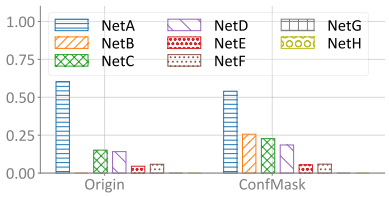


Figure 7: Clustering coefficients of all anonymized networks, with $k_R = 6$ and $k_H = 2$

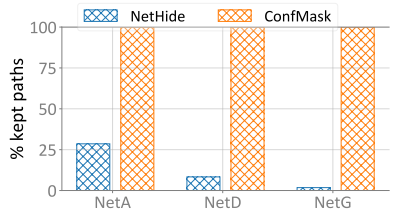


Figure 8: Proportion of exactly kept paths

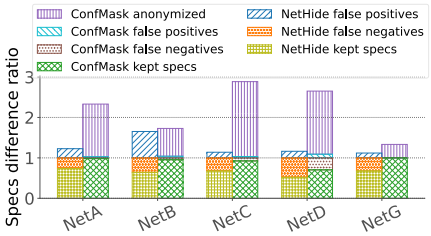


Figure 9: Preserved network specifications via Config2Spec, with $k_R = 6$ and $k_H = 4$

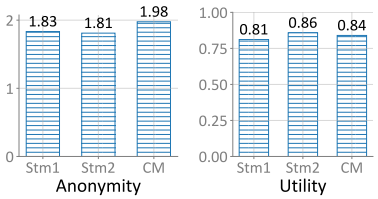


Figure 10: Anonymity and utility comparison

7 Evaluation

We evaluate ConfMask on several networks under a variety of parameter settings, to address the key questions below:

Q1: How does ConfMask preserve privacy and utility? By a default setting of $k_H = 2$, $k_R = 6$, and a random noise generator, our solution can achieve on average 1.93 times better privacy metrics, while ensuring functional equivalence.

Q2: How do different parameters affect the results of ConfMask, and what is the tradeoff? We observed that both k_H and k_R negatively affect the configuration utility metric U_C to different extents, and there is a moderate negative correlation between U_C and the route anonymity metric N_r, with a correlation coefficient of -0.36.

Q3: How efficiently does ConfMask perform and scale up to large networks? We compare ConfMask with 2 strawman solutions, and show that our solution scales up to large networks well by outperforming strawman 2 by around 85% and handling all large test networks within minutes.

Datasets: There are 8 groups of network configurations in our evaluation (Table 2), including OSPF-only networks and BGP+OSPF networks, with sizes ranging from 18 to 219. Among them, networks A, B and C use real-world configuration files, while the configurations of networks D ~ H are auto-generated by scripts from network topologies available online[24], which include some of the largest network topologies we can obtain for now.

7.1 Effectiveness of Privacy and Utility

To evaluate the effectiveness of ConfMask on achieving privacy and utility, we measure the following five metrics: *(a) Route Anonymity*: number of distinct routing paths between edge routers N_r; *(b) Route Utility*: percentage of exactly kept host-to-host paths; *(c) Topology Anonymity*: minimum number of nodes sharing the same degree k_d; *(d) Topology Utility*: Clustering Coefficient of network topology,

ID	Network	$\|R\|$	$\|H\|$	$\|E\|$	#config lines	Network Type
A	Enterprise	10	8	26	1095	BGP+OSPF
B	University	13	8	25	1652	BGP+OSPF
C	Backbone	11	9	22	980	BGP+OSPF
D	Bics	49	98	162	4410	OSPF
E	Columbus	86	68	169	6968	OSPF
F	USCarrier	161	58	378	13940	OSPF
G	FatTree04	20	16	48	1544	OSPF
H	FatTree08	72	64	320	8448	OSPF

Table 2: The evaluation networks

a metric widely used in graph anonymization works [25]; and *(e) Configuration Utility*: depending on the number of lines injected into configuration files N_l and the total number of lines P_l, configuration utility is defined as $U_C = 1 - N_l/P_l$.

Route anonymity and utility: We use the minimum and average N_r to indicate how much a routing path can be hidden among others, thus adding up to the difficulty of distinguishing the real routing paths. As shown in Figure 5, our solution can reach the average anonymity of 1.93 routing paths per pair of routers, by setting the number of fake hosts added $k_H = 2$. We also verify that ConfMask fulfills SFE conditions (i.e., preserve all original FIB entries) by ensuring our system terminates on all test networks.

Topology anonymity and utility: We also evaluate the anonymity and structural utility achieved by ConfMask from the topology perspective. To be consistent with the topology k-anonymity definition above, we measure the minimum number of nodes that sharing the same degree in the graph, and present the result in Figure 6. Since ConfMask ensures k-anonymity by design, the minimum k is always larger than or equal to the input argument k_R, regardless of the topology structure of the original network.

To make a network built on top of a topology meet the route utility properties that ConfMask promises, the anonymized topology should at least have all original edges and nodes preserved.

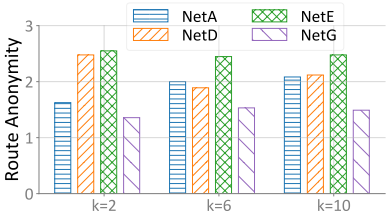


Figure 11: Impact of router degree (k_R) on route anonymity (N_r)

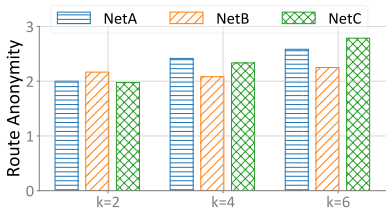


Figure 12: Impact of host degree (k_H) on route anonymity (N_r)

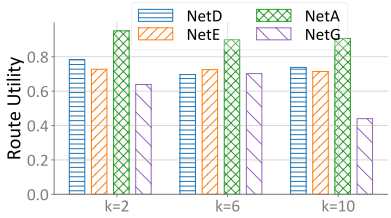


Figure 13: Impact of router degree (k_R) on config utility (U_C)

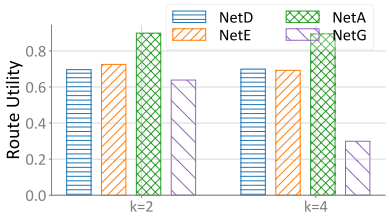


Figure 14: Impact of host degree (k_H) on config utility (U_C)

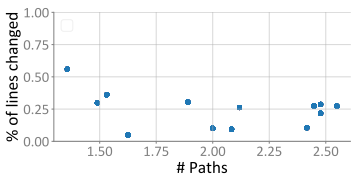


Figure 15: Route anonymity (N_r) versus config utility (U_C)

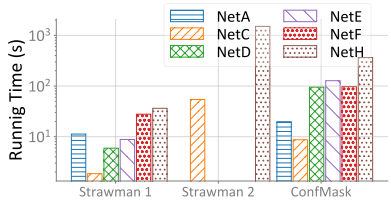


Figure 16: Running time comparison

Since ConfMask leverages incremental anonymization algorithm, the preservation of existing nodes and edges is guaranteed. We evaluate a stronger utility standard Clustering Coefficient (CC). Figure 7 shows the comparison of CC value between anonymized graphs and the original one. ConfMask maintains a similar structure to the original topology, with an average difference of only 0.075.

Configuration utility: In addition to the similarity of the configuration files and data plane of the anonymized and original networks, downstream tasks of ConfMask such as collaborative network troubleshooting, highly rely on the anonymized configuration files being similar to the original files, which we measure by the number of lines modified during the anonymization process. Our tests show that, on average, our solution only needs to add ~25% of lines to the configuration files, and on large scale network, the percentage can goes down to ~5%. The configuration breakdown can be found in Appendix Table 3.

7.2 Route Anonymity Algorithms

We compare ConfMask with NetHide [30] and two strawman approaches (introduced in §4.3) on the following metrics: (a) Route Utility: the percentage of exactly preserved host-to-host paths P_U and network specifications [7]; (b) Route Anonymity: N_r, and (c) Configuration Utility: U_C.

First, by measuring P_U, we show that NetHide fails to keep all host-to-host paths exactly, which means it could not preserve our definition of Functional Equivalence. As is shown in Fig. 8, in every test network, NetHide can only preserve less than 30% of paths exactly, with an average of ~15%, and down to ~1%. While our solution can guarantee 100% preservation by ensuring SFE.

In addition, we compare the differences in network specifications before and after anonymization with Config2Spec [7], which includes `Reachability`, `Waypoint` and `LoadBalance`), to show that NetHide fails to preserve original forwarding behavior. A network specification consists of a set of policies, each capturing a specific behavior in the network (e.g., reachability of two routers). In Fig. 9, the "kept spec" bars show that NetHide only keeps 65.2% of original specifications, while ConfMask preserves 91.3% specifications on average, reducing the number of missing specifications by 75%. The bars above 1 show that ConfMask can introduce 3.55 times more anonymization specifications than NetHide, while 96.9% of the introduced specifications by ConfMask are for the new fake hosts and links, so that no significant false positive specifications are introduced to the network.

We then compare ConfMask with strawman solutions discussed previously to show their effect or drawbacks on route privacy and configuration utility. Figure 10(L) compares N_r across 2 strawman solutions and ConfMask. Results show that ConfMask slightly outperforms two strawman solutions, with average N_r of 1.98, 1.83, and 1.81 respectively. However, the strawman approaches either expose patterns for easy de-anonymization (§4.3) or take too much running time (§7.3).

We also compare our solution with strawman solutions from the utility perspective. Figure 10(R) shows that, Strawman 1, by filtering all hosts on all fake interfaces, injects a significantly larger amount of code lines into the configuration files, reaching 21.2% more than ConfMask. On the other hand, although strawman 2 is conservative in adding filters, achieving average 13.1% fewer lines of code injected than ConfMask, but it suffers from a small pace, resulting in extra long running time.

7.3 Parameter Sensitivity and Performance

In this section, we review the experiment result to show the trade-off between privacy and utility metrics, as well as the effect of each parameter to the overall metrics of ConfMask outputs. We also show that ConfMask improved the anonymization performance compared to two strawman solutions.

Anonymity parameters: Figure 11-14 shows individual impacts of k_R and k_H on route anonymity and utility metrics. We found

that, topology parameter k_R doesn't really affect route anonymity(Figure 11), fixing $k_H = 2$ and setting k_R to $2, 6, 10$ results in average N_r of $2.00, 1.97, 2.04$, revealing no strong correlation. On the contrary, N_r is largely decided on k_H, i.e., the number of fake hosts corresponding to each real hosts in the original network. When fixing $k_R = 6$, most N_r grows as k_H increases. Stepping k_H in $2, 4, 6$ results in average N_r of 2.05, 2.29, 2.54. However, both parameters negatively impacting the configuration utility. As k_R grows from 2 to 10, U_C can drop by 1% to 20%, while as k_H grows from 2 to 6, U_C drops moderately by 0% to 3%.

Privacy-utility trade-off: To quantitatively study the trade-off between route privacy and utility, we plot each test case as a data point in the number of distinct paths against percentage of configuration lines injected. Figure 15 shows that, the two metrics are loosely negatively correlated, with $r = -0.36$.

Performance evaluation: We care about the performance of ConfMask, so that it can scale efficiently to larger networks. Figure 16 shows that strawman 1 performs the fastest, with 60% to tens of times less end-to-end running time than ConfMask, sacrificing the privacy. Strawman 2 runs slowest among all solutions, due to the high computation complexity of inspecting routes one by one. It takes average 8 to 100 times running time than ConfMask, incurring unacceptable time cost. While for ConfMask, it can finish anonymizing the largest network, FatTree-08, in around 6 minutes.

8 Related Work

Graph anonymization: Most existing approaches in topology anonymization focus on satisfying certain anonymity targets like k-anonymity [19, 25, 26, 45], k-automorphism [46], or k-isomorphism [11], but they mainly focus on social networks and lack utility consideration in computer networks. Moreover, they consider only unlabeled graphs, meaning that additional information such as IP addresses and configurations can still make the network topology insecure.

Routing anonymization: Another type of approach aims at defending network threats such as link-flooding attacks (LFA) by misleading adversaries to get a wrong network topology. For instance, HoneyNet [23] deceives and detects attackers by redirecting traceroute packets, NetHide [30] diverts the data paths from the control paths, and EqualNet [22] further adds virtual nodes in addition to virtual links. However, these approaches reduce the utility of legitimate use, and may require programmable devices in the network which are not widely applicable nowadays. Moreover, these obfuscations still leak implicit privacy information and are not desirable when precise utility properties are required.

Trace anonymization: Network traces contain a lot of sensitive information that can indicate individual users' activities. `tcpmkpub` [36] is a tool for modifying sensitive attributes to realize trace anonymization, but has been shown vulnerable to topology inference attacks [14]. Another approach is to realize differential privacy on the network trace data [28]. However, high privacy levels can lead to low fidelity results, which is undesirable for cases requiring accurate data for analyses. There is also a flow-based trace anonymization strategy, (k, j)-obfuscation [37], which guarantees a high-level privacy via obfuscating sensitive data in network flows.

IP Anonymization: There has been work based on cryptographic means to anonymize IP addresses in files. The earliest work on this approach is `TCPdpriv` [31], a table-based IP address anonymizer. Later, Crypto-Pan [39] was extended as a cryptographic sanitization tool of network trace data including IP addresses in a prefix-preserving manner. Such a method is also extended to large scale distributed setting [43]. ConfMask is compatible with these techniques but focuses on implicit information anonymization.

9 Discussion

Internet hosts: In this work, we only take into consideration the internal hosts which are configured inside the network. It is possible to also include hosts outside of given network configurations, for example Internet hosts. One potential way is to divide Internet destinations into routing Equivalent Classes as often used in network verification [5], and substitute the concept of *host* with Equivalent Classes. We will leave this extension to future works.

PII obfuscation: We do not propose PII obfuscation as the critical stage of ConfMask's anonymization pipeline, but ConfMask is compatible with any text-based information obfuscation technique as downstream plug-in tasks. ConfMask generates configuration files that follow the same syntax as the input files. Therefore, existing cryptographic IP anonymization and BGP AS Number hashing [27], and password hashing [21] works are all applicable to the output of ConfMask as privacy enhancement.

Network scale obfuscation: Although we implement ConfMask with a basic graph anonymization algorithm and do not consider the number of routers to be a key attribute to hide in the anonymization process, our theoretical proof of functional equivalence does not require the set of routers to remain unchanged before and after anonymization. The workflow of modifying topology, and correcting and obfuscating routing paths is also applicable as long as the graph algorithm does not remove existing routers. Therefore, theoretically, ConfMask is extendable with graph anonymization algorithms that modify the number of nodes (e.g., [12, 41]) to enable the obfuscation of the number of routers. However, the anonymity definition of router number and how to auto-generate new configuration files for the additional routers while keeping them indistinguishable from the human-configured routers are challenging problems to address. We leave these extensions to future works.

10 Conclusion

Sharing network configuration files can provide significant benefits for both network management and research purposes. However, the potential privacy risks associated with sharing such files must be carefully addressed. In this paper, we present ConfMask as a privacy-preserving configuration sharing tool that effectively anonymizes topology and routing behaviors while preserving functional equivalence. Our evaluation shows that ConfMask scales well for large networks and achieves the desired level of anonymity.

Acknowledgments

We would like to thank our shepherd Ennan Zhai and the anonymous SIGCOMM reviewers for their constructive comments. This work was supported by Peking University startup funding and NYU Shanghai startup funding. Guyue Liu is the corresponding author.

References

[1] [n. d.]. The Internet2 Project. ([n. d.]). https://internet2.edu/ https://internet2.edu/.

[2] 2011. Cisco BGP question about advertising routes. https://serverfault.com/questions/257000/cisco-bgp-question-about-advertising-routes

[3] 2015. Cisco BGP Wrong Next Hop. https://networkengineering.stackexchange.com/questions/16873/cisco-bgp-wrong-next-hop

[4] 2023. Cisco triangle network with static routing not working. https://networkengineering.stackexchange.com/questions/83133/cisco-triangle-network-with-static-routing-not-working

[5] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2017. A General Approach to Network Configuration Verification. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, Los Angeles, CA, USA, 155–168. https://doi.org/10.1145/3098822.3098834

[6] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2018. Control Plane Compression. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, Budapest Hungary, 476–489. https://doi.org/10.1145/3230543.3230583

[7] Rüdiger Birkner, Dana Drachsler-Cohen, Laurent Vanbever, and Martin Vechev. 2020. {Config2Spec}: Mining Network Specifications from Network Configurations. 969–984. https://www.usenix.org/conference/nsdi20/presentation/birkner

[8] Paolo Boldi, Francesco Bonchi, Aris Gionis, and Tamir Tassa. 2012. Injecting Uncertainty in Graphs for Identity Obfuscation. *Proceedings of the VLDB Endowment* 5, 11 (2012), 1376–1387. https://doi.org/10.48550/ARXIV.1208.4145

[9] Francesco Bonchi, Aristides Gionis, and Tamir Tassa. 2014. Identity Obfuscation in Graphs Through the Information Theoretic Lens. *Information Sciences* 275 (Aug. 2014), 232–256. https://doi.org/10.1016/j.ins.2014.02.035

[10] Justin Brickell and Vitaly Shmatikov. 2008. The Cost of Privacy: Destruction of Data-Mining Utility in Anonymized Data Publishing. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Las Vegas, Nevada, USA, 70–78. https://doi.org/10.1145/1401890.1401904

[11] James Cheng, Ada Wai-chee Fu, and Jia Liu. 2010. k-Isomorphism: Privacy Preserving Network Publication Against Structural Attacks. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, Indianapolis, Indiana, USA, 459–470. https://doi.org/10.1145/1807167.1807218

[12] Sean Chester, Bruce Kapron, Ganesh Ramesh, Gautam Srivastava, Alex Thomo, and Venkatesh Srinivasan. 2011. k-Anonymization of Social Networks by Vertex Addition *(CEUR Workshop Proceedings, Vol. 789)*. 107–116.

[13] Cisco U. by Learning and Certifications. 2023. Network Troubleshooting Made Easy with ChatGPT | Snack Minute Ep. 96. https://youtu.be/W7KtYF0gDJU https://youtu.be/W7KtYF0gDJU.

[14] Scott E. Coull, Charles V. Wright, Fabian Monrose, Michael P. Collins, and Michael K. Reiter. 2007. Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Network Traces. In *NDSS*, Vol. 7. 35–47.

[15] Ahmed El-Hassany, Petar Tsankov, Laurent Vanbever, and Martin Vechev. 2018. NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI '18)*. 579–594.

[16] Ari Fogel, Stanley Fung, Luis Pedrosa, Meg Walraed-Sullivan, Ramesh Govindan, Ratul Mahajan, and Todd Millstein. 2015. A General Approach to Network Configuration Analysis. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI '15)*. 469–483.

[17] Damien Garros. 2020. The State of Network Operation Through Automation / NetDevOps Survey 2019. https://blog.networktocode.com/post/state-network-operations-netdevops-survey-2019/

[18] T.G. Griffin, F.B. Shepherd, and G. Wilfong. 2002. The Stable Paths Problem and Interdomain Routing. *IEEE/ACM Transactions on Networking* 10, 2 (April 2002), 232–243. https://doi.org/10.1109/90.993304

[19] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. 2007. Anonymizing Social Networks. *Computer Science Department Faculty Publication Series* (Jan. 2007), 180.

[20] Brett Holbert, Srikar Tati, Simone Silvestri, Thomas F. La Porta, and Ananthram Swami. 2015. Network Topology Inference With Partial Information. *IEEE Transactions on Network and Service Management* 12, 3 (Sept. 2015), 406–419. https://doi.org/10.1109/TNSM.2015.2451032

[21] Intentionet. 2023. Netconan - A Network Configuration Anonymizer. (April 2023). v https://internet2.edu/.

[22] Jinwoo Kim, Eduard Marin, Mauro Conti, and Seungwon Shin. 2022. EqualNet: A Secure and Practical Defense for Long-Term Network Topology Obfuscation. In *Network and Distributed Systems Security (NDSS) Symposium 2022*. San Diego, CA, USA. https://doi.org/10.14722/ndss.2022.23154

[23] Jinwoo Kim and Seungwon Shin. 2017. Software-Defined HoneyNet: Towards Mitigating Link Flooding Attacks. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, Denver, CO, USA, 99–100. https://doi.org/10.1109/DSN-W.2017.10

[24] Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. 2011. The Internet Topology Zoo. *IEEE Journal on Selected Areas in Communications* 29, 9 (Oct. 2011), 1765–1775. https://doi.org/10.1109/JSAC.2011.111002

[25] Kun Liu and Evimaria Terzi. 2008. Towards Identity Anonymization on Graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*. ACM Press, Vancouver, Canada, 93. https://doi.org/10.1145/1376616.1376629

[26] Xuesong Lu, Yi Song, and Stéphane Bressan. 2012. Fast Identity Anonymization on Graphs. In *International Conference on Database and Expert Systems Applications*. Springer, Berlin, Heidelberg, 281–295.

[27] David A. Maltz, Jibin Zhan, Geoffrey Xie, Hui Zhang, Gísli Hjálmtýsson, Albert Greenberg, and Jennifer Rexford. 2004. Structure Preserving Anonymization of Router Configuration Data. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. ACM, Taormina Sicily, Italy, 239–244. https://doi.org/10.1145/1028788.1028819

[28] Frank McSherry and Ratul Mahajan. 2010. Differentially-Private Network Trace Analysis. *ACM SIGCOMM Computer Communication Review* 40, 4 (Aug. 2010), 123–134. https://doi.org/10.1145/1851275.1851199

[29] Frank D. McSherry. 2009. Privacy Integrated Queries: An Extensive Platform for Privacy-Preserving Data Analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, Providence Rhode Island USA, 19–30. https://doi.org/10.1145/1559845.1559850

[30] Roland Meier, Petar Tsankov, Vincent Lenders, Laurent Venbever, and Martin Vechev. 2018. NetHide: Secure and Practical Network Topology Obfuscation. In *27th USENIX Security Symposium (USENIX Security 18)*. 693–709.

[31] Greg Minshall. 1997. TCPdpriv. (1997). https://ita.ee.lbl.gov/html/contrib/tcpdpriv.html https://ita.ee.lbl.gov/ html/contrib/tcpdpriv.html.

[32] Meisam Mohammady, Lingyu Wang, Yuan Hong, Habib Louafi, Makan Pourzandi, and Mourad Debbabi. 2018. Preserving Both Privacy and Utility in Network Trace Anonymization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Toronto Canada, 459–474. https://doi.org/10.1145/3243734.3243809

[33] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust De-anonymization of Large Sparse Datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, Oakland, CA, USA, 111–125. https://doi.org/10.1109/SP.2008.33 ISSN: 1081-6011.

[34] Jason C. Neumann. 2015. *The Book of GNS3: Build Virtual Network Labs using Cisco, Juniper, and More*. No Starch Press.

[35] OpenAI. [n. d.]. OpenAI Privacy policy. ([n. d.]). https://openai.com/policies/privacy-policy https://openai.com/ policies/privacy-policy.

[36] Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. 2006. The Devil and Packet Trace Anonymization. *ACM SIGCOMM Computer Communication Review* 36, 1 (Jan. 2006), 29–38. https://doi.org/10.1145/1111322.1111330

[37] Daniele Riboni, Antonio Villani, Domenico Vitali, Claudio Bettini, and Luigi V. Mancini. 2012. Obfuscation of Sensitive Data in Network Flows. In *2012 Proceedings IEEE INFOCOM*. 2372–2380. https://doi.org/10.1109/INFCOM.2012.6195626

[38] Joseph Severini, Radhika Niranjan Mysore, Vyas Sekar, Sujata Banerjee, and Michael K. Reiter. 2021. The Netivus Manifesto: Making Collaborative Network Management Easier for the Rest of us. *ACM SIGCOMM Computer Communication Review* 51, 2 (April 2021), 10–17. https://doi.org/10.1145/3464994.3464997

[39] Adam Slagell, Jun Wang, and William Yurcik. 2004. Network Log Anonymization: Application of Crypto-Pan to Cisco Netflows. In *Proceedings of the Workshop on Secure Knowledge Management 2004*.

[40] Latanya Sweeney. 2002. k-Anonymity: A Model For Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (Oct. 2002), 557–570. https://doi.org/10.1142/S0218488502001648

[41] Nazanin Takbiri, Xiaozhe Shao, Lixin Gao, and Hossein Pishro-Nik. 2019. Improving Privacy in Graphs Through Node Addition. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, Monticello, IL, USA, 487–494. https://doi.org/10.1109/ALLERTON.2019.8919967

[42] Samuel T. Trassare, Robert Beverly, and David Alderson. 2013. A Technique for Network Topology Deception. In *2013 IEEE Military Communications Conference (MILCOM 2013)*. 1795–1800. https://doi.org/10.1109/MILCOM.2013.303

[43] Jun Xu, Jinliang Fan, Mostafa H. Ammar, and Sue B. Moon. 2002. Prefix-Preserving IP Address Anonymization: Measurement-Based Security Evaluation and a New Cryptography-Based Scheme. In *10th IEEE International Conference on Network Protocols, 2002. Proceedings*. IEEE Comput. Soc, Paris, France, 280–289. https://doi.org/10.1109/ICNP.2002.1181415

[44] Xian Zhang and Chris Phillips. 2012. A Survey on Selective Routing Topology Inference Through Active Probing. *IEEE Communications Surveys & Tutorials* 14, 4 (2012), 1129–1141. https://doi.org/10.1109/SURV.2011.081611.00040

[45] Bin Zhou and Jian Pei. 2008. Preserving Privacy in Social Networks Against Neighborhood Attacks. In *2008 IEEE 24th International Conference on Data Engineering*. Cancun, Mexico, 506–515. https://doi.org/10.1109/ICDE.2008.4497459

[46] Lei Zou, Lei Chen, and M. Tamer Özsu. 2009. k-Automorphism: A General Framework for Privacy Preserving Network Publication. *Proceedings of the VLDB Endowment* 2, 1 (Aug. 2009), 946–957. https://doi.org/10.14778/1687627.1687734

Appendices are supporting material that has not been peer-reviewed.

A Proof that SFE Conditions Imply Functional Equivalence

Our algorithm aims to achieve functional equivalence via fulfilling the SFE (strong functional equivalence) conditions. To this end, we show that the SFE conditions are sufficient to guarantee functional equivalence, for distance-vector protocols, link-state protocols, and the BGP protocol (Theorem A.4). We may first recall and mathematically formulate functional equivalence and SFE conditions as follows.

Functional equivalence

$\exists f$ injective, such that:

(1) $\big(f(r) \in \mathcal{A}(r),\ \forall r \in R\big) \ \wedge\ \big(f(h) \in \mathcal{A}(h),\ \forall h \in H\big)$;

(2) $\big((h_s, r_1, \cdots, r_n, h_d) \in DP\big) \iff \big((\widehat{h_s}, \widehat{r_1}, \cdots, \widehat{r_n}, \widehat{h_d}) \in \widehat{DP}\big)$, where $\big(\widehat{h_s} = f(h_s),\ \widehat{h_d} = f(h_d),\ \widehat{r_i} = f(r_i),\ \forall i\big)$.

SFE conditions *Assume that no existing router is removed during anonymization, i.e., $R \subseteq \mathcal{A}(R)$.*

Distance-vector protocols

(1) $\big(e = (r, r') \in E_R\big) \implies \big(\widehat{e} = (\mathcal{A}(r), \mathcal{A}(r')) \in \mathcal{A}(E_R)\big) \ \wedge\ \big(\widehat{e}$ has the same link properties as $e\big)$;
$\big(e = (r, h) \in E_H\big) \implies \big(\widehat{e} = (\mathcal{A}(r), \mathcal{A}^0(h)) \in \mathcal{A}(E_H)\big) \ \wedge\ \big(\widehat{e}$ has the same link properties as $e\big)$.

(2) $\big(\widehat{e} = (\widehat{v}, \widehat{v'}) \in \mathcal{A}(E)\big) \ \wedge\ \big(\widehat{v}$ has some route imported from $\widehat{v'}\big) \implies \big(e = (\mathcal{A}^{-1}(\widehat{v}), \mathcal{A}^{-1}(\widehat{v'})) \in E\big)$.

BGP

(1) Each AS satisfies the SFE conditions for distance-vector protocols.

(2) The network of ASes satisfies the SFE conditions for distance-vector protocols.

Link-state protocols

(1) $\big(e = (r, r') \in E_R\big) \implies \big(\widehat{e} = (\mathcal{A}(r), \mathcal{A}(r')) \in \mathcal{A}(E_R)\big) \ \wedge\ \big(\texttt{cost}(\widehat{e}) = \texttt{cost}(e)\big)$;
$\big(e = (r, h) \in E_H\big) \implies \big(\widehat{e} = (\mathcal{A}(r), \mathcal{A}^0(h)) \in \mathcal{A}(E_H)\big) \ \wedge\ \big(\texttt{cost}(\widehat{e}) = \texttt{cost}(e)\big)$.

(2) $\big(\widehat{e} = (\widehat{v}, \widehat{v'}) \in \mathcal{A}(E)\big) \ \wedge\ \big(e = \mathcal{A}^{-1}(v), \mathcal{A}^{-1}(v') \notin E\big)$
$\implies \big(\texttt{cost}(\widehat{e}) > \texttt{min\_cost}(\mathcal{A}^{-1}(v), \mathcal{A}^{-1}(v'))\big) \ \vee\ \big(\texttt{cost}(\widehat{e}) = \texttt{min\_cost}(\mathcal{A}^{-1}(v), \mathcal{A}^{-1}(v')) \ \wedge\ \widehat{e}$ is rejected$\big)$.

Lemma A.1. Suppose that $\widehat{CFG}$ follows a distance-vector routing protocol. If $\widehat{CFG}$ satisfies the SFE conditions for distance-vector protocols with respect to CFG, then $CFG \overset{F}{\simeq} \widehat{CFG}$.

Proof. In order to prove functional equivalence, we need to find an injective mapping f that satisfies its definition. Note that we have assumed $R \subseteq \mathcal{A}(R)$, so we can choose f as the identity mapping when acting on R. As for hosts, the original hosts are preserved in the anonymized network though fake hosts may be added. For each original host $h \in H$, recall that $\mathcal{A}^0$ maps it to the corresponding unique real host in the anonymized network, so we choose $f = \mathcal{A}^0$ when acting on H. Clearly $f(r) \in \mathcal{A}(r)$ for all $r \in R$ and $f(h) \in \mathcal{A}(h)$ for all $h \in H$. Also note that this injective mapping f we choose does not map to fake hosts, so when inversely mapping hosts in the anonymized network back to hosts in the original network, we do not consider fake hosts and routing paths involving fake hosts. Apart from these, it suffices to show that the forwarding behavior in the dataplane between the original network and the anonymized network mapped via f are identical.

Showing that $DP \subseteq \widehat{DP}$: Fix an arbitrary routing path $p = (h_s, r_1, \cdots, r_n, h_d) \in DP$, forwarding from h_s to h_d in the original network, then $(h_s, r_1) \in E$, $(r_1, r_2) \in E$, $\cdots$, $(r_n, h_d) \in E$. By *Condition 1*, this implies that

$$\begin{aligned}
&(f(h_s), f(r_1)) \in \mathcal{A}(E), && \text{having the same link properties as } (h_s, r_1),\\
&(f(r_1), f(r_2)) \in \mathcal{A}(E), && \text{having the same link properties as } (r_1, r_2),\\
&\quad\cdots && \quad\cdots\\
&(f(r_n), f(h_d)) \in \mathcal{A}(E), && \text{having the same link properties as } (r_n, h_d).
\end{aligned}$$

This forms a path $\widehat{p} = (f(h_s), f(r_1), \cdots, f(r_n), f(h_d))$, but we assume for contradiction that $\widehat{p} \notin \widehat{DP}$. The only possibility for this is that, there exists another routing path $\widetilde{p} = (f(h_s), \widetilde{r_1}, \cdots, \widetilde{r_{n'}}, f(h_d)) \in \widehat{DP}$, such that $\widetilde{p}$ is more preferred than $\widehat{p}$ as the routing path from $f(h_s)$ to $f(h_d)$ in the anonymized network. Since $\widetilde{p} \in \widehat{DP}$, this means that

$$\begin{aligned}
&(f(h_s), \widetilde{r_1}) \in \mathcal{A}(E), && \text{with } f(h_s) \text{ having some route imported from } \widetilde{r_1},\\
&(\widetilde{r_1}, \widetilde{r_2}) \in \mathcal{A}(E), && \text{with } \widetilde{r_1} \text{ having some route imported from } \widetilde{r_2},\\
&\quad\cdots && \quad\cdots\\
&(\widetilde{r_{n'}}, f(h_d)) \in \mathcal{A}(E), && \text{with } \widetilde{r_{n'}} \text{ having some route imported from } f(h_d).
\end{aligned}$$

Then by *Condition 2*, we can see that $(h_s, f^{-1}(\widetilde{r_1})) \in E$, $(f^{-1}(\widetilde{r_1}), f^{-1}(\widetilde{r_2})) \in E$, $\cdots$, $(f^{-1}(\widetilde{r_{n'}}), h_d) \in E$. This would form a path $p' = (h_s, f^{-1}(\widetilde{r_1}), \cdots, f^{-1}(\widetilde{r_{n'}}), h_d)$ in the original network. Again by *Condition 1*, we have that

$$\begin{aligned}
&(f(h_s), \widetilde{r_1}) \text{ has the same link properties as } (h_s, f^{-1}(\widetilde{r_1})),\\
&(\widetilde{r_1}, \widetilde{r_2}) \text{ has the same link properties as } (f^{-1}(\widetilde{r_1}), f^{-1}(\widetilde{r_2})),\\
&\qquad \cdots\\
&(\widetilde{r_{n'}}, f(h_d)) \text{ has the same link properties as } (f^{-1}(\widetilde{r_{n'}}), h_d).
\end{aligned}$$

Therefore, since $\widetilde{p}$ is preferred over $\widehat{p}$ as the routing path from $f(h_s)$ to $f(h_d)$ in the anonymized network, p' must be preferred over p as the routing path from h_s to h_d in the original network as well. Then $p \notin DP$, leading to a contradiction. Therefore, we have shown that for an arbitrary routing path $p = (h_s, r_1, \cdots, r_n, h_d) \in DP$, its mapped version $\widehat{p} = (f(h_s), f(r_1), \cdots, f(r_n), f(h_d))$ must be a routing path in the anonymized network as well.

Showing that $DP \supseteq \widehat{DP}$**:** As is said, we do not consider routing paths that involve fake hosts. Fix an arbitrary routing path $\widehat{p} = (\widehat{h_s}, \widehat{r_1}, \cdots, \widehat{r_{n'}}, \widehat{h_d}) \in \widehat{DP}$, forwarding from $\widehat{h_s}$ to $\widehat{h_d}$ in the anonymized network. Note that we have $f^{-1} = \mathcal{A}^{-1}$, so that

$$\begin{aligned}
&(\widehat{h_s}, \widehat{r_1}) \in \mathcal{A}(E), && \text{with } \widehat{h_s} \text{ having some route imported from } \widehat{r_1},\\
&(\widehat{r_1}, \widehat{r_2}) \in \mathcal{A}(E), && \text{with } \widehat{r_1} \text{ having some route imported from } \widehat{r_2},\\
&\qquad \cdots && \qquad \cdots\\
&(\widehat{r_{n'}}, \widehat{h_d}) \in \mathcal{A}(E), && \text{with } \widehat{r_{n'}} \text{ having some route imported from } \widehat{h_d}.
\end{aligned}$$

Then by *Condition 2*, we can see that $(f^{-1}(\widehat{h_s}), f^{-1}(\widehat{r_1})) \in E$, $(f^{-1}(\widehat{r_1}), f^{-1}(\widehat{r_2})) \in E$, $\cdots$, $(f^{-1}(\widehat{r_{n'}}), f^{-1}(\widehat{h_d})) \in E$. This forms a path $p = (f^{-1}(\widehat{h_s}), f^{-1}(\widehat{r_1}), \cdots, f^{-1}(\widehat{r_{n'}}), f^{-1}(\widehat{h_d}))$, but we assume for contradiction that $p \notin DP$. The only possibility for this is that, there exists another routing path $\underline{p} = (f^{-1}(\widehat{h_s}), \underline{r_1}, \cdots, \underline{r_n}, f^{-1}(\widehat{h_d})) \in DP$, such that $\underline{p}$ is more preferred than p as the routing path from $f^{-1}(\widehat{h_s})$ to $f^{-1}(\widehat{h_d})$ in the original network. Since $\underline{p} \in DP$, we have that $(f^{-1}(\widehat{h_s}), \underline{r_1}) \in E$, $(\underline{r_1}, \underline{r_2}) \in E$, $\cdots$, $(\underline{r_n}, f^{-1}(\widehat{h_d})) \in E$. Then by *Condition 1*, this implies that

$$\begin{aligned}
&(\widehat{h_s}, f(\underline{r_1})) \in \mathcal{A}(E), && \text{having the same link properties as } (f^{-1}(\widehat{h_s}), \underline{r_1}),\\
&(f(\underline{r_1}), f(\underline{r_2})) \in \mathcal{A}(E), && \text{having the same link properties as } (\underline{r_1}, \underline{r_2}),\\
&\qquad \cdots && \qquad \cdots\\
&(f(\underline{r_n}), \widehat{h_d}) \in \mathcal{A}(E), && \text{having the same link properties as } (\underline{r_n}, f^{-1}(\widehat{h_d})).
\end{aligned}$$

This would form a path $\widehat{\underline{p}} = (\widehat{h_s}, f(\underline{r_1}), \cdots, f(\underline{r_n}), \widehat{h_d})$, with each link on $\widehat{\underline{p}}$ having the same link properties as each link on $\underline{p} \in DP$. Therefore, since $\underline{p}$ is preferred over p as the routing path from $f^{-1}(\widehat{h_s})$ to $f^{-1}(\widehat{h_d})$ in the original network, $\widehat{\underline{p}}$ must be preferred over $\widehat{p}$ in the anonymized network as well. Then $\widehat{p} \notin \widehat{DP}$, leading to a contradiction. Therefore, we have shown that for an arbitrary routing path $\widehat{p} = (\widehat{h_s}, \widehat{r_1}, \cdots, \widehat{r_{n'}}, \widehat{h_d}) \in \widehat{DP}$, its inversly mapped version $p = (f^{-1}(\widehat{h_s}), f^{-1}(\widehat{r_1}), \cdots, f^{-1}(\widehat{r_{n'}}), f^{-1}(\widehat{h_d}))$ must be a routing path in the original network as well.

Concluding the proof: For the specific f that we have chosen,

$$f(v) = \begin{cases} v, & \text{if } v \in R,\\ \mathcal{A}^0(v), & \text{if } v \in H. \end{cases}$$

Moreover, not considering routing paths that involve fake hosts, we have shown that the dataplanes DP in the original network and $\widehat{DP}$ in the anonymized network are identical, i.e., an arbitrary path $(h_s, r_1, \cdots, r_n, h_d)$ is a routing path in the original network if and only if $(\widehat{h_s}, \widehat{r_1}, \cdots, \widehat{r_n}, \widehat{h_d})$ is a routing path in the anonymized network, with $\widehat{h_s} = f(h_s)$, $\widehat{h_d} = f(h_d)$, and $\widehat{r_i} = f(r_i)$ for all $i = 1, \cdots, n$. This concludes that $CFG \overset{F}{\simeq} \widehat{CFG}$, so the proof is complete. □

Lemma A.2. Suppose that $\widehat{CFG}$ follows BGP. If $\widehat{CFG}$ satisfies the SFE conditions for link state protocols with respect to CFG, then $CFG \overset{F}{\simeq} \widehat{CFG}$.

Proof. This is trivial. Choose the same f as in Lemma A.1 Within each AS, *Condition 1* ensures functional equivalence by Lemma A.1, and thus all routers, hosts, links, and routing behaviors remain unchanged. On top of that, *Condition 2* again ensures functional equivalence for the network of ASes by Lemma A.1, thus the inter-AS routing behaviors are also preserved. The proof is thus complete. □

Lemma A.3. Suppose that $\widehat{CFG}$ follows a link-state routing protocol. If $\widehat{CFG}$ satisfies the SFE conditions for link state protocols with respect to CFG, then $CFG \overset{F}{\simeq} \widehat{CFG}$.

Proof. We choose f in the exact same way as in Lemma A.1. Again, not consider the fake hosts and routing paths involving fake hosts, it then suffices to show that the forwarding behavior in the dataplane between the original network and the anonymized network mapped via f are identical.

Showing that $DP \subseteq \widehat{DP}$**:** Fix an arbitrary routing path $p = (h_s, r_1, \cdots, r_n, h_d) \in DP$, forwarding from h_s to h_d in the original network, then $(h_s, r_1) \in E$, $(r_1, r_2) \in E$, $\cdots$, $(r_n, h_d) \in E$. By *Condition 1*, this implies that

$$\begin{aligned}
&(f(h_s), f(r_1)) \in \mathcal{A}(E), && \texttt{cost}((f(h_s), f(r_1))) = \texttt{cost}((h_s, r_1)),\\
&(f(r_1), f(r_2)) \in \mathcal{A}(E), && \texttt{cost}((f(r_1), f(r_2))) = \texttt{cost}((r_1, r_2)),\\
&\quad\cdots && \quad\cdots\\
&(f(r_n), f(h_d)) \in \mathcal{A}(E), && \texttt{cost}((f(r_n), f(h_d))) = \texttt{cost}((r_n, h_d)).
\end{aligned}$$

This forms a path $\widehat{p} = (f(h_s), f(r_1), \cdots, f(r_n), f(h_d))$, but we assume for contradiction that $\widehat{p} \notin \widehat{DP}$. The only possibility for this is that, there exists another routing path $\widetilde{p} = (f(h_s), \widetilde{r_1}, \cdots, \widetilde{r_{n'}}, f(h_d)) \in \widehat{DP}$, such that $\widetilde{p}$ is more preferred than $\widehat{p}$ as the routing path from $f(h_s)$ to $f(h_d)$ in the anonymized network. Since $\widetilde{p} \in \widehat{DP}$, this means that

$$\begin{aligned}
&\widetilde{e_0} = (f(h_s), \widetilde{r_1}) \in \mathcal{A}(E), && \text{with } \texttt{cost}(\widetilde{e_0}) = \texttt{min_cost}(f(h_s), \widetilde{r_1}) \text{ and } \widetilde{e_0} \text{ is not rejected},\\
&\widetilde{e_1} = (\widetilde{r_1}, \widetilde{r_2}) \in \mathcal{A}(E), && \text{with } \texttt{cost}(\widetilde{e_1}) = \texttt{min_cost}(\widetilde{r_1}, \widetilde{r_2}) \text{ and } \widetilde{e_1} \text{ is not rejected},\\
&\quad\cdots && \quad\cdots\\
&\widetilde{e_{n'}} = (\widetilde{r_{n'}}, f(h_d)) \in \mathcal{A}(E), && \text{with } \texttt{cost}(\widetilde{e_{n'}}) = \texttt{min_cost}(\widetilde{r_{n'}}, f(h_d)) \text{ and } \widetilde{e_{n'}} \text{ is not rejected}.
\end{aligned}$$

The reason is, if any of $\widetilde{e_i}$, $i = 0, \cdots, n'$ is rejected, then clearly $\widetilde{p}$ cannot be a valid routing path. Moreover, if any of $\widetilde{e_i}$, $i = 0, \cdots, n'$ does not have its cost as the minimum cost between its source and destination, there exists another link $\widetilde{e^*}$, such that $\texttt{cost}(\widetilde{e^*}) < \texttt{cost}(\widetilde{e_i})$ for some $i = 1, \cdots, n'$. Then by simply replacing $\widetilde{e_i}$ with $\widetilde{e^*}$ in $\widetilde{p}$, we will be able to obtain a new path from $f(h_s)$ to $f(h_d)$ with an overall lower cost than $\widetilde{p}$, contradicting $\widetilde{p} \in \widehat{DP}$. Now we can use the contrapositive of *Condition 2*. Since $\widetilde{e_i} \in \mathcal{A}(E)$ is already guaranteed to be true for all $i = 0, \cdots, n'$, it can only be the case that $f^{-1}(\widetilde{e_i}) \notin E$ is false for all $i = 0, \cdots, n'$, so that $e_0 = (h_s, f^{-1}(\widetilde{r_1})) \in E$, $e_1 = (f^{-1}(\widetilde{r_1}), f^{-1}(\widetilde{r_2})) \in E$, $\cdots$, $e_{n'} = (f^{-1}(\widetilde{r_{n'}}), h_d) \in E$. Moreover, by *Condition 1*, we have that $\texttt{cost}(e_i) = \texttt{cost}(\widetilde{e_i})$ for all $i = 0, \cdots, n'$. Therefore, since $\widetilde{p}$ is preferred over $\widehat{p}$ as the routing path from $f(h_s)$ to $f(h_d)$ in the anonymized network, p' must be preferred over p as the routing path from h_s to h_d in the original network as well. Then $p \notin DP$, leading to a contradiction. Therefore, we have shown that for an arbitrary routing path $p = (h_s, r_1, \cdots, r_n, h_d) \in DP$, its mapped version $\widehat{p} = (f(h_s), f(r_1), \cdots, f(r_n), f(h_d))$ must be a routing path in the anonymized network as well.

Showing that $DP \supseteq \widehat{DP}$**:** As is said, we do not consider routing paths that involve fake hosts. Fix an arbitrary routing path $\widehat{p} = (\widehat{h_s}, \widehat{r_1}, \cdots, \widehat{r_{n'}}, \widehat{h_d}) \in \widehat{DP}$, forwarding from $\widehat{h_s}$ to $\widehat{h_d}$ in the anonymized network. Note that we have $f^{-1} = \mathcal{A}^{-1}$, so that

$$\begin{aligned}
&\widehat{e_0} = (\widehat{h_s}, \widehat{r_1}) \in \mathcal{A}(E), && \text{with } \texttt{cost}(\widehat{e_0}) = \texttt{min_cost}(\widehat{h_s}, \widehat{r_1}) \text{ and } \widehat{e_0} \text{ is not rejected},\\
&\widehat{e_1} = (\widehat{r_1}, \widehat{r_2}) \in \mathcal{A}(E), && \text{with } \texttt{cost}(\widehat{e_1}) = \texttt{min_cost}(\widehat{r_1}, \widehat{r_2}) \text{ and } \widehat{e_1} \text{ is not rejected},\\
&\quad\cdots && \quad\cdots\\
&\widehat{e_{n'}} = (\widehat{r_{n'}}, \widehat{h_d}) \in \mathcal{A}(E), && \text{with } \texttt{cost}(\widehat{e_{n'}}) = \texttt{min_cost}(\widehat{r_{n'}}, \widehat{h_d}) \text{ and } \widehat{e_{n'}} \text{ is not rejected}.
\end{aligned}$$

The reason is, if any of $\widehat{e_i}$, $i = 0, \cdots, n'$ is rejected, then clearly $\widehat{p}$ cannot be a valid routing path. Moreover, if any of $\widehat{e_i}$, $i = 0, \cdots, n'$ does not have its cost as the minimum cost between its source and destination, there exists another link $\widehat{e^*}$, such that $\texttt{cost}(\widehat{e^*}) < \texttt{cost}(\widehat{e_i})$ for some $i = 1, \cdots, n'$. Then by simply replacing $\widehat{e_i}$ with $\widehat{e^*}$ in $\widehat{p}$, we will be able to obtain a new path from $\widehat{h_s}$ to $\widehat{h_d}$ with an overall lower cost than $\widehat{p}$, contradicting $\widehat{p} \in \widehat{DP}$. Then we can use the contrapositive of *Condition 2*. Since $\widehat{e_i} \in \mathcal{A}(E)$ is already guaranteed to be true for all $i = 0, \cdots, n'$, it can only be the case that $f^{-1}(\widehat{e_i}) \in E$ is false for all $i = 0, \cdots, n'$, so that $e_0 = (f^{-1}(\widehat{h_s}), f^{-1}(\widehat{r_1})) \in E$, $e_1 = (f^{-1}(\widehat{r_1}), f^{-1}(\widehat{r_2})) \in E$, $\cdots$, $e_{n'} = (f^{-1}(\widehat{r_{n'}}), f^{-1}(\widehat{h_d})) \in E$. This forms a path $p = (f^{-1}(\widehat{h_s}), f^{-1}(\widehat{r_1}), \cdots, f^{-1}(\widehat{r_{n'}}), f^{-1}(\widehat{h_d}))$, but we assume for contradiction that $p \notin DP$. The only possibility for this is that, there exists another routing path $\underline{p} = (f^{-1}(\widehat{h_s}), \underline{r_1}, \cdots, \underline{r_n}, f^{-1}(\widehat{h_d})) \in DP$, such that $\underline{p}$ is more preferred than p as the routing path from $f^{-1}(\widehat{h_s})$ to $f^{-1}(\widehat{h_d})$ in the original network. Since $\underline{p} \in DP$, this means that $\underline{e_0} = (f^{-1}(\widehat{h_s}), \underline{r_1}) \in E$, $\underline{e_1} = (f^{-1}(\widehat{r_1}), \underline{r_2}) \in E$, $\cdots$, $\underline{e_n} = (f^{-1}(\widehat{r_n}), \underline{h_d}) \in E$. Then by *Condition 1*, this implies that

$$\begin{aligned}
&\widehat{\underline{e_0}} = (\widehat{h_s}, f(\underline{r_1})) \in \mathcal{A}(E), && \text{with } \texttt{cost}(\widehat{\underline{e_0}}) = \texttt{min_cost}(\widehat{h_s}, f(\underline{r_1})),\\
&\widehat{\underline{e_1}} = (f(\underline{r_1}), f(\underline{r_2})) \in \mathcal{A}(E), && \text{with } \texttt{cost}(\widehat{\underline{e_1}}) = \texttt{min_cost}(f(\underline{r_1}), f(\underline{r_2})),\\
&\quad\cdots && \quad\cdots\\
&\widehat{\underline{e_n}} = (f(\underline{r_n}), \widehat{h_d}) \in \mathcal{A}(E), && \text{with } \texttt{cost}(\widehat{\underline{e_n}}) = \texttt{min_cost}(f(\underline{r_n}), \widehat{h_d}).
\end{aligned}$$

This would form a path $\widehat{\underline{p}} = (\widehat{h_s}, f(\underline{r_1}), \cdots, f(\underline{r_n}), \widehat{h_d})$, with each link on $\widehat{\underline{p}}$ having the same cost as each link on $\underline{p} \in DP$, i.e., $\texttt{cost}(\underline{e_i}) = \texttt{cost}(\widehat{\underline{e_i}})$ for all $i = 0, \cdots, n$. Therefore, since $\underline{p}$ is preferred over p as the routing path from $f^{-1}(\widehat{h_s})$ to $f^{-1}(\widehat{h_d})$ in the original network, $\widehat{\underline{p}}$

must be preferred over $\widehat{p}$ in the anonymized network as well. Then $\widehat{p} \notin \widehat{DP}$, leading to a contradiction. Therefore, we have shown that for an arbitrary routing path $\widehat{p} = (\widehat{h_s}, \widehat{r_1}, \cdots, \widehat{r_{n'}}, \widehat{h_d}) \in \widehat{DP}$, its inversely mapped version $p = (f^{-1}(\widehat{h_s}), f^{-1}(\widehat{r_1}), \cdots, f^{-1}(\widehat{r_{n'}}), f^{-1}(\widehat{h_d}))$ must be a routing path in the original network as well.

Concluding the proof: We have chosen f same as in Lemma A.1. Not considering routing paths that involve fake hosts, we have shown that the dataplanes DP in the original network and $\widehat{DP}$ in the anonymized network are identical, i.e., an arbitrary path $(h_s, r_1, \cdots, r_n, h_d)$ is a routing path in the original network if and only if $(\widehat{h_s}, \widehat{r_1}, \cdots, \widehat{r_n}, \widehat{h_d})$ is a routing path in the anonymized network, with $\widehat{h_s} = f(h_s)$, $\widehat{h_d} = f(h_d)$, and $\widehat{r_i} = f(r_i)$ for all $i = 1, \cdots, n$. This concludes that $SRP \overset{F}{\cong} \widehat{SRP}$, so the proof is complete. □

Theorem A.4. If $\widehat{CFG}$ satisfies the SFE conditions with respect to CFG under some distance-vector routing protocol, BGP, or some link-state routing protocol, then $CFG \overset{F}{\cong} \widehat{CFG}$.

Proof. This immediately follows from Lemma A.1, Lemma A.2, and Lemma A.3 by choosing the injective mapping

$$f(v) = \begin{cases} v, & \text{if } v \in R, \\ \mathcal{A}^0(v), & \text{if } v \in H. \end{cases}$$

It concludes that SFE conditions implies functional equivalence for our supported types of protocols. □

B Proof that Functional Equivalence Preserves Routing Utilities

Now that we have shown that the SFE conditions imply functional equivalence. In other words, the anonymized networks generated by our algorithm automatically satisfies functional equivalence. To see why this is useful, we will show that functional equivalence preserves various desired utility properties (Theorem B.7). We start by rigorously restating the definitions of the important routing utility properties listed in §3.

Routing utility property definitions:

(1) *Reachability*: There exists a routing path from $h_s \in H$ to $h_d \in H$ in the original network *if and only if* there exists a routing path from $\widehat{h_s} \in \widehat{H}$ to $\widehat{h_d} \in \widehat{H}$ in the anonymized network as well, for some $\widehat{h_s} \in \mathcal{A}(h_s)$ and $\widehat{h_d} \in \mathcal{A}(h_d)$.
(2) *Path-lengths*: All the routing paths from $h_s \in H$ to $h_d \in H$ in the original network have length l *if and only if* all the routing paths from $\widehat{h_s} \in \widehat{H}$ to $\widehat{h_d} \in \widehat{H}$ in the anonymized network have length l as well, for some $\widehat{h_s} \in \mathcal{A}(h_s)$ and $\widehat{h_d} \in \mathcal{A}(h_d)$.
(3) *Black-holes*: Traffic sent from h_s to h_d is dropped along some path $\{h_s, r_1, \cdots, r_n\} \in DP$ in the original network *if and only if* traffic sent from $\widehat{h_s}$ to $\widehat{h_d}$ is dropped along the path $\{\widehat{h_s}, \widehat{r_1}, \cdots, \widehat{r_n}\} \in \widehat{DP}$ in the anonymized network, with $\widehat{h_s} \in \mathcal{A}(h_s)$, $\widehat{h_d} \in \mathcal{A}(h_d)$ and $\widehat{r_i} \in \mathcal{A}(r_i)$ for all $i = 1, \cdots, n$.
(4) *Multipath-consistency*: Traffic sent from $h_s \in H$ is reachable along some routing path to $h_d \in H$ but dropped along another routing path in the original network *if and only if* traffic sent from $\widehat{h_s} \in \widehat{H}$ is reachable along some routing path to $\widehat{h_d} \in \widehat{H}$ but dropped along another routing path in the anonymized network, with $\widehat{h_s} \in \mathcal{A}(h_s)$ and $\widehat{h_d} \in \mathcal{A}(h_d)$.
(5) *Waypointing*: Traffic is waypointed through one of the routers among $\{r_1, \cdots, r_n\} \in 2^R$ in the original network *if and only if* traffic is waypointed through one of the routers among $\{\widehat{r_1}, \cdots, \widehat{r_n}\} \in 2^{\mathcal{A}(R)}$ in the anonymized network as well, with $\widehat{r_i} \in \mathcal{A}(r_i)$ for all $i = 1, \cdots, n$.
(6) *Routing-loops*: Traffic from h_s to h_d enters a routing loop in G *if and only if* traffic from $\widehat{h_s}$ to $\widehat{h_d}$ enters a routing loop in $\widehat{G}$ as well.

Lemma B.1. If $CFG \overset{F}{\cong} \widehat{CFG}$, then the anonymized network preserves reachability from the original network.

Proof. To see that reachability is preserved, we need to show that for any pair of hosts $(h_s, h_d) \in H \times H$, h_d is reachable from h_s *if and only if* $\widehat{h_d}$ is reachable from $\widehat{h_s}$, for some $\widehat{h_s} \in \mathcal{A}(h_s)$ and $\widehat{h_d} \in \mathcal{A}(h_d)$. Assume that $CFG \overset{F}{\cong} \widehat{CFG}$.

($\Rightarrow$) Fix arbitrary h_s and h_d and assume that h_d is reachable from h_s, then there exists a routing path $p = (h_s, r_1, \cdots, r_k, h_d) \in DP$. By functional equivalence, there exists an injective mapping f, such that $f(h_s) \in \mathcal{A}(h_s)$, $f(h_d) \in \mathcal{A}(h_d)$, $f(r_i) \in \mathcal{A}(r_i)$ for all $i = 1, \cdots, k$, and $\widehat{p} = (f(h_s), f(r_1), \cdots, f(r_k), f(h_d))$ is a routing path in the anonymized network. Therefore, $f(h_d)$ is also reachable $f(h_s)$. Since h_s and h_d are taken arbitrarily, and $f(h_s) \in \mathcal{A}(h_s)$, $f(h_d) \in \mathcal{A}(h_d)$, this direction is done.

($\Leftarrow$) This direction is symmetric to above since functional equivalence is an *if and only if* condition. □

Lemma B.2. If $CFG \overset{F}{\cong} \widehat{CFG}$, then the anonymized network preserves path-lengths from the original network.

Proof. To see that path-lengths are preserved, we need to show that for any pair of hosts $(h_s, h_d) \in H \times H$, all routing paths from h_s to h_d are of length l *if and only if* all routing paths from $\widehat{h_s}$ to $\widehat{h_d}$ are of length l, for some $\widehat{h_s} \in \mathcal{A}(h_s)$ and $\widehat{h_d} \in \mathcal{A}(h_d)$. This is equivalent to showing the contrapositive, that is, there exists a routing path from h_s to h_d not of length l *if and only if* there exists a routing path from $\widehat{h_s}$ to $\widehat{h_d}$ not of length l, with $\widehat{h_s} = \mathcal{A}(h_s)$ and $\widehat{h_d} = \mathcal{A}(h_d)$. Assume that $CFG \overset{F}{\cong} \widehat{CFG}$.

($\Rightarrow$) Fix arbitrary h_s and h_d and assume that there exists a routing path from h_s to h_d of length $k + 2 \neq l$, which we denote by $p = (h_s, r_1, \cdots, r_k, h_d) \in DP$. By functional equivalence, there exists an injective mapping f, such that $f(h_s) \in \mathcal{A}(h_s)$, $f(h_d) \in \mathcal{A}(h_d)$, $f(r_i) \in \mathcal{A}(r_i)$ for all $i = 1, \cdots, k$, and $\widehat{p} = (f(h_s), f(r_1), \cdots, f(r_k), f(h_d))$ is a routing path in the anonymized network. Clearly the length

of $\widehat{p}$ is also $k+2 \neq l$. Since h_s ad h_d are taken arbitrarily, and $f(h_s) \in \mathcal{A}(h_s)$, $f(h_d) \in \mathcal{A}(h_d)$, this direction is done.

($\Leftarrow$) This direction is symmetric to above since functional equivalence is an *if and only if* condition. □

Lemma B.3. If $CFG \overset{F}{\cong} \widehat{CFG}$, then the anonymized network preserves black-holes from the original network.

Proof. *next_hop* to destination host h_d on router r is in the original network $\Leftrightarrow$ $\widehat{next\_hop}$ to destination host $\widehat{h_d}$ on router $\widehat{r}$ is in the anonymized network. So if traffic sent from h_s to h_d is dropped because h_d is in the routing table of r_{n-1} but not in the routing table of next hop r_n, in the anonymized network, traffic sent from $\widehat{h_s}$ to $\widehat{h_d}$ will also be dropped because the same traffic will pass router $\widehat{r_{n-1}}$, and $\widehat{h_d}$ is in the routing table of $\widehat{r_{n-1}}$ but not in the next hop $\widehat{r_n}$. □

Lemma B.4. If $CFG \overset{F}{\cong} \widehat{CFG}$, then the anonymized network preserves multipath-consistency from the original network.

Proof. If multiple paths p, p' exist in the original network from h_s to h_d, and there exists some traffic reachable through p and dropped along p', in the anonymized network, $\widehat{p}$ and $\widehat{p'}$ are both present in the dataplane, so some traffic will be reachable through $\widehat{p}$ and dropped through $\widehat{p'}$. □

Lemma B.5. If $CFG \overset{F}{\cong} \widehat{CFG}$, then the anonymized network preserves waypointing from the original network.

Proof. To see that waypointing is preserved, we need to show that traffic is waypointed through one of $\{r_1, \cdots, r_n\} \in 2^R$ in the original network *if and only if* traffic is waypointed through one of $\{\widehat{r_1}, \cdots, \widehat{r_n}\} \in 2^{\mathcal{A}(R)}$ in the anonymized network as well, with $\widehat{r_i} \in \mathcal{A}(r_i)$ for all $i = 1, \cdots, n$. Assume that $CFG \overset{F}{\cong} \widehat{CFG}$ and take an arbitrary set $\mathcal{R} = \{r_1, \cdots, r_n\} \in 2^R$.

($\Rightarrow$) Assume that traffic is waypointed through some $r \in \mathcal{R}$, which means that there exists a path $p = (h_s, \cdots, r, \cdots, h_d) \in DP$, including r on its way. By functional equivalence, there exists an injective mapping f, such that $f(h_s) \in \mathcal{A}(h_s)$, $f(h_d) \in \mathcal{A}(h_d)$, and $f(r) \in \mathcal{A}(r)$, and $\widehat{p} = (f(h_s), \cdots, f(r), \cdots, f(h_d))$ is a routing path in the anonymized network. Clearly traffic would be waypointed through $f(r) \in \mathcal{A}(r)$, so this direction is done.

($\Leftarrow$) This direction is symmetric to above since functional equivalence is an *if and only if* condition. □

Lemma B.6. If $CFG \overset{F}{\cong} \widehat{CFG}$, then the anonymized network preserves routing-loops from the original network.

Proof. Some traffic sent from h_s to h_d enters a routing loop $\Rightarrow$ the chain of *next_hop* points to a router it has already passed through. As SFE conditions preserve the *next_hop* of host destinations on each router, the corresponding routing loop exists in the anonymized network. □

Theorem B.7. If $CFG \overset{F}{\cong} \widehat{CFG}$, then $\widehat{CFG}$ preserves reachability, path-lengths, black-holes, multipath-consistency, waypointing, and routing-loops from CFG.

Proof. This immediately follows from Lemma B.1, Lemma B.2, Lemma B.3, Lemma B.4, Lemma B.5, and Lemma B.6. It concludes that functional equivalence preserves all our defined routing utility properties and thus produces a usable anonymized network. □

C Evaluation Results

Here we attach Table 3, present the full evaluation result of configuration utility(U_C). As is analyzed in the paper, k_H and k_R has moderate negative impacts on U_C, thus leading the # of added lines goes up as they increase. FatTree-08 has zero added interface lines due to the special structure of network (many subgraphs are isomorphism, and nodes share the same degrees).

Network, Parameters	#Added Routing Protocol Lines	#Added Filter Lines	#Added Interface Lines	# Total Lines
BICS, k_R=2, k_H=2	84	822	16	4250
BICS, k_R=6, k_H=2	144	987	160	4250
BICS, k_R=6, k_H=4	131	1408	96	5438
BICS, k_R=10, k_H=2	124	894	96	4250
Columbus, k_R=2, k_H=2	155	1820	8	7270
Columbus, k_R=6, k_H=2	162	1812	32	7300
Columbus, k_R=6, k_H=4	159	2317	32	8159
Columbus, k_R=10, k_H=2	179	1870	80	7440
CCNP, k_R=2, k_H=2	21	42	12	1503
CCNP, k_R=6, k_H=2	45	85	42	1696
CCNP, k_R=6, k_H=4	47	91	42	1706
CCNP, k_R=10, k_H=2	39	68	42	1587
FatTree-08, k_R=2, k_H=2	503	4742	0	9364
FatTree-08, k_R=6, k_H=2	433	1859	0	6341
FatTree-08, k_R=6, k_H=4	512	10004	0	15004
FatTree-08, k_R=10, k_H=2	385	1313	0	5699
USCarrier, k_R=6, k_H=2	341	7334	32	19419

Table 3: # of lines modified in each component by ConfMask, compared to the total number of lines in configuration files.