Exploiting Path Diversity for Thwarting Pollution Attacks in Named Data Networking

Haoran Guo, *Student Member, IEEE,* Xiaodong Wang, Kun Chang, and Ye Tian, *Member, IEEE*

***Abstract*—With information becoming a first-class citizen on the Internet, Information-centric Networking (ICN) is considered as a promising direction for the future Internet. Named Data Networking (NDN) is a prominent example of emerging ICN architectures. Unfortunately, NDN is vulnerable to various attacks targeting its in-network caching mechanism. In this paper, we focus on the false-locality pollution attack, in which an adversary repeatedly requests a number of unpopular data objects to waste the precious cache space on the NDN router and to reduce normal users' hit ratios. With simulation experiments, we show that such an attack can cause considerable damage to the NDN network. To detect and mitigate such an attack, we introduce an algorithm that exploits the diversity of the Interest traversing paths within an ISP's point-of-presence (PoP) network. We also propose inexpensive methodologies based on the probabilistic counting and Bloom filter techniques to implement the algorithm on an NDN router. The experimental results indicate that our proposed algorithm is effective in thwarting false-locality pollution. We also experiment with strategies that the adversary may utilize against our anti-pollution algorithm and demonstrate that such strategies are either ineffective or impractical in the real world.**

***Index Terms*—Future Internet architecture, cache pollution attack, network security**

I. Introduction

With information becoming a first-class citizen on the Internet, various Information-centric Networking (ICN) architectures have been proposed [1]. Among them, Named Data Networking (NDN) [2] [3] is a prominent instance that has attracted growing interest from academic and industrial research communities in recent years.

In contrast to the current Internet, which is based on addresses, the main building blocks of NDN are *named data objects*, and NDN changes the semantics of network services from delivering the packet to a given destination address to fetching the object identified by a given name [3]. To enable this, the NDN architecture's thin-waist layer provides the universal functionalities of routing, forwarding, and (optional) caching of the named data objects on network nodes. More specifically, each data object under NDN has a URL-like name that is globally unique. When a router receives an object-requesting message (named *Interest* in NDN), it queries its *Pending Interest Table* (PIT) for the pending Interests for the same data. If there are pending Interests, the newly arrived Interest is aggregated with them; otherwise, the router looks up its *Forwarding Information Base* (FIB) using the object's name and forwards the request to the appropriate interfaces toward the content servers that serve the data. The pending Interests are kept in the PIT until the corresponding data object (in NDN's *Data* message) traverses the node along the reverse path or times out. However, when the router has already cached the object in its local *Content Store*, it can directly respond to the Interest with its cached replica.

NDN employs an in-network caching mechanism, where a router may cache any traversing data objects and use them to satisfy future requests. However, decades of Internet experience suggests that such a networked cache system is vulnerable under *poisoning* and *pollution* attacks. In a poisoning attack, the adversary may exploit a large number of compromised hosts (zombies) and poison the network by injecting fake objects into the cache. For example, DNS spoofing [4] is an attack that poisons caches on local DNS servers. In a pollution attack, the adversary injects irrelevant but legitimate objects to waste precious cache resources to reduce users' hit ratios and satisfaction levels. A well-known example is the pollution attack launched by the music industry against peer-to-peer file sharing systems [5].

There are primarily two types of cache pollution attacks: *locality-disruption pollution* and *false-locality pollution* [6]. In locality-disruption pollution, an attacker requests a large number of unpopular objects to weaken the content locality in the cache; in contrast, a false-locality pollution attacker repeatedly requests a few unpopular objects to build false localities, thus misleading the system to cache these objects and waste precious cache space.

As a future Internet architecture, NDN is expected to be resilient against existing and potential attacks by design; however, unfortunately, NDN is vulnerable to various types of attacks targeting its in-network caching mechanism [7] [8]. In recent years, people have begun to examine potential poisoning and pollution threats and enhance the architecture in different ways [9] [10] [11]. In this paper, we focus on the false-locality pollution attack. We exploit the diversity of the Interest traversing paths within an ISP's point-of-presence (PoP) network and present an algorithm for detecting and mitigating the pollution. More specifically, we make the following contributions in this paper:

- We propose a unified model to profile an attacker's strategy for polluting an NDN network. We show with simulation experiments that, by carefully selecting his strategy parameter, a false-locality pollution attacker can cause considerable damage with limited resources on the NDN network.

The authors are with the Anhui Key Laboratory on High Performance Computing and Application, School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230026, P. R. China. E-mail: {hrguo, qsmywxd, curtis}@mail.ustc.edu.cn, yetian@ustc.edu.cn.

Corresponding author: Ye Tian (yetian@ustc.edu.cn)

- We analyze ISPs' PoP networks with real-world Internet measurement datasets and show that, for popular data objects, there exist diverse Interest traversing paths within an NDN PoP network.
- We design an algorithm that exploits the diversity of the Interest traversing paths within a PoP for detecting and mitigating the pollution. We theoretically analyze the rationale of the algorithm and present two approaches based on the probabilistic counting and Bloom filter techniques to implement the algorithm on an NDN backbone router.
- We evaluate our proposed anti-pollution algorithm with simulation experiments. We show that the algorithm is effective in thwarting a pollution attack without incurring significant overhead. Moreover, we experiment on two strategies that the adversary may take against the anti-pollution algorithm and find that they are either ineffective or impractical. We also discuss the issue of how to decide the algorithm parameters when the adversary's strategy is unknown.

The remainder part of this paper is organized as follows. Section II surveys and discusses the related works. We analyze the NDN PoP network in Section IV. In Section V, we describe and analyze our proposed anti-pollution algorithm and evaluate it's performance in Section VI. Finally, we conclude this paper in Section VII.

II. Related Work

As a promising future Internet architecture, NDN [2] [3] has attracted considerable attention from academia and industry and has rapidly become a hot topic in the network research community in recent years. Although NDN seeks to provide data security with its built-in primitives (*i.e.*, cryptographically signing data objects by producer), the architecture might be vulnerable to various attacks. For example, So *et al.* [12] consider a hash flooding denial-of-service (DoS) attack in which an attacker maliciously introduces hash collisions to degrade an NDN router's performance. Virgilio *et al.* [13] show that the PIT component in an NDN router is vulnerable to a distributed DoS (DDoS) attack that seeks to exhaust the PIT's memory with forged Interests.

As an architecture with in-network caches, NDN is threatened by cache poisoning and pollution attacks. Mauri *et al.* [14] consider a case in which an attacker exploits NDN to gather a massive amount of storage for malicious contents. Ghali *et al.* [8] analyze the root cases of content poisoning attacks in NDN networks. Li *et al.* [9] propose a lightweight mechanism for integrity verification and access control against poisoning attacks on NDN networks. Xie *et al.* [10] study the locality-disruption pollution attack and propose a mechanism named CacheShield to make the NDN network more robust by increasing the attack cost. Conti *et al.* [11] show that proactive approaches, such as CacheShield, are not effective and propose to passively detect the locality-disruption attack by sampling and comparing the Interest distributions on the NDN router. Karami *et al.* [15] propose to detect locality-disruption and false-locality pollution attacks with an artificial neural network named ANFIS. However, there are several drawbacks to the method. First, the neural network must be trained in advance given that the attacker's polluting behaviors are already known. Second, the neural network contains five layers of over thirty adaptive nodes, with each node performing considerable computations, and the neural network is driven with detailed statistics of the cached data objects such as the mean and variance of the access frequencies in a number of recent intervals. Considering the heavy computation overhead of the neural network, such a method is impractical for implementation as an online detecting algorithm for an NDN router because the router needs to update its Content Store at line speed.

The work that is perhaps the most similar to ours is [6], where the authors study locality-disruption and false-locality attacks against file cache systems in IP networks. The basic idea for detecting the false-locality pollution is that the system traces files requested by clients. When a file is requested by a small set of clients repeatedly, the system is considered as being exploited by pollution attackers, and the file is evicted. This solution also suffers from drawbacks. First, under NDN, clients request named data objects rather than files, and the former's granularity is substantially smaller than that of the latter. There will be considerable overhead for an NDN router to trace all the requesting clients for each cached data object. Moreover, in [6], clients are identified with their IP addresses. However, in both IP and NDN networks, an attacker can easily forge source addresses on compromised hosts to bypass the detection. In our proposed anti-pollution algorithm in this paper, we trace the diversity of the Interest traversing paths for each cached object, which incurs moderate overhead in the NDN router, and the path information is generated by routers hop by hop along the path. Thus, it is difficult to forge the information on compromised hosts.

III. Pollution attacks in NDN

In this section, we develop a unified model to profile a pollution attacker's attacking strategy and examine the effectiveness of the attack in NDN networks. As in previous works [6] [10] [11] [15], we assume that an attacker has the knowledge of some unpopular contents on the Internet. The unpopular contents could be news reports from many years ago, mailing list achieves from many years ago, web pages from websites that rank low on Alexa, user-generated videos that have few clicks on YouTube, etc. Note that it is infeasible to prevent pollution by maintaining a complete list of unpopular contents on the Internet.

A. Threat model and attack strategy

We consider a pollution attack whereby an adversary compromises one or multiple hosts in an NDN network and uses them to request a set $\mathcal{P}$ of unpopular objects. The attack seeks to inject these objects in $\mathcal{P}$ into the NDN router's Content Store and reduce the normal users' hit ratios.

We employ two parameters to profile the attacker's strategy. The first parameter is the attacker's attacking power, which determines how many Interest messages he is capable of sending toward the NDN router. We use the ratio between

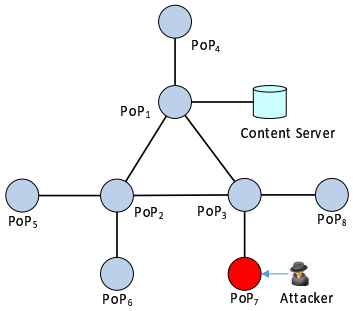


Fig. 1. PoP topology of network in simulation.

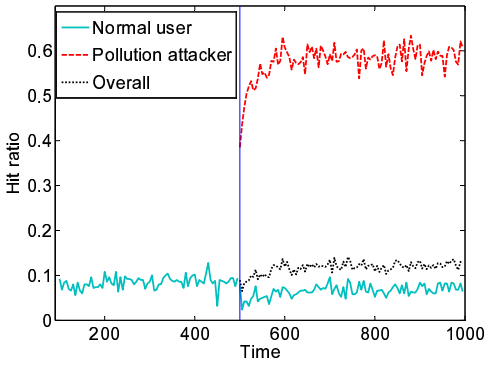


Fig. 2. Instantaneous hit ratios before and after the pollution attack is launched at time 500 for $\gamma = 10\%$ and $\theta = 0.6$.

the malicious Interest messages sent out by the attacker and the Interests sent by all the clients to describe the attacking power and refer to it as the attacker's *power ratio*, denoted as γ.

The second parameter is the data object set $\mathcal{P}$ that the attacker exploits for pollution. We use the term *range ratio* θ, which is the ratio between $\mathcal{P}$'s cardinality and the router's cache space C, to describe the attacker's object selection range, *i.e.*, $\theta = \frac{|\mathcal{P}|}{C}$. Moreover, for each individual data object in $\mathcal{P}$, the attacker may select unpopular objects to maximize the pollution effectiveness. On the other hand, the attacker may request some moderately popular contents, which are not popular enough to be cached by the router but which continue to be requested by normal users, to make the requests more difficult to distinguish from normal users' requests. We use the *accumulative popularity* ρ of the objects in $\mathcal{P}$ to describe the attacker's choices, *i.e.*, $\rho = \sum_{e \in \mathcal{P}} p(e)$, where $p(e)$ is object e's popularity.

Note that by modeling the pollution strategy as $\{\gamma, \theta, \rho\}$, we can describe a wide range of attacking behaviors. For example, given a limited power ratio constraint γ, if an attacker has a large θ value, in other words, he requests a large number of data objects for polluting the NDN router, this is a locality-disruption pollution attack because the attacker cannot request each of the objects in $\mathcal{P}$ with sufficient frequency. On the other hand, if the attacker has a moderate θ, it is easier for him to build false localities for the objects in $\mathcal{P}$ and launch a false-locality pollution attack. In this paper, we only consider typical false-locality pollution attacks with $\theta \leq 1$.

B. Effectiveness of pollution attack

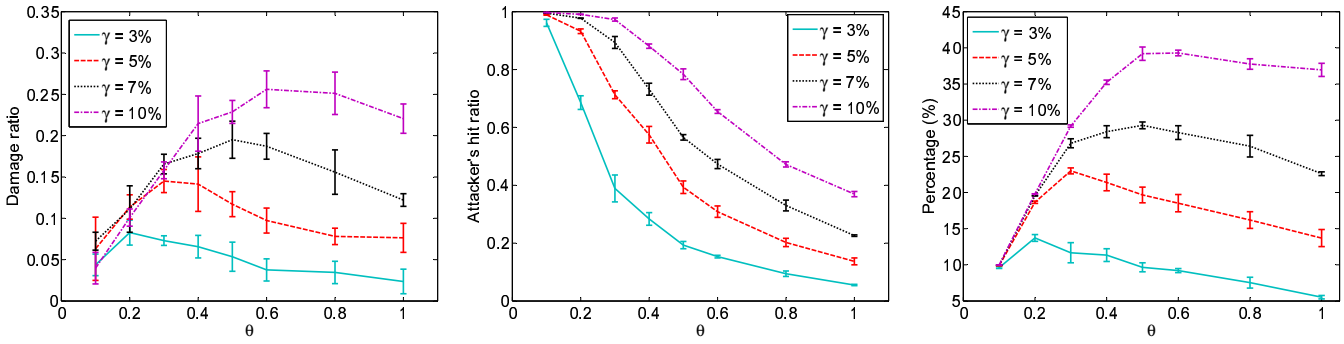
We examine the effectiveness of the false-locality pollution attack with simulation experiments. Fig. 1 presents the network topology used in our simulation, where each node represents an ISP's point of presence (PoP), and users connect to routers in these PoPs to request and consume data objects. For simplicity, in this section, we temporarily assume that each PoP has only one fully functional NDN router that is directly connected to users, and all the data objects are originally served by the content server at PoP_1. The network employs the "leave a copy everywhere" (*LCE*) strategy to place data replicas at all the downstream routers on cache hits [2], and each NDN router applies the "least frequently used" (*LFU*) policy for evicting cached replicas. We choose LFU because LFU is more vulnerable to false-locality pollution attacks [6]; thus, with LFU, the pollution damage and the effectiveness of our proposed algorithm for thwarting the pollution will be more pronounced.

We simulate a content catalog with one million data objects and use the Mandelbrot-Zipf (MZipf) distribution, which is widely observed for web and peer-to-peer traffics [16] [17], to model data object popularity. More specifically, under the MZipf model, for the k^{th} most-popular object, its probability of being requested is

$$p(k) = \frac{1/(k+q)^{\alpha}}{\sum_i 1/(i+q)^{\alpha}} \tag{1}$$

where α ($\alpha > 0$) is the model's skewness factor, which determines how fast the popularity decays as the rank increases, and q ($q > 0$) is referred to as the plateau factor. If we plot an MZipf mass distribution curve under a log-log scale, the higher the value of α is, the steeper the distribution curve will be; the higher the value of q is, the more flattened the head of the curve will be. In the following experiments, we always set $\alpha = 0.8$ and $q = 10$. The NDN router's cache space is fixed as 0.1% of the entire content catalog.

We first consider an attacker with only one compromised host in PoP_7. The attacker has an attacking capacity of $\gamma = 10\%$ and sets his range ratio as $\theta = 0.6$. We assume that the attacker only requests the least popular objects in the content catalog to maximize the pollution effectiveness. Fig. 2 presents the instantaneous hit ratios achieved by normal users before and after the attack is launched at time 500; we also plot the attacker's hit ratio and the overall hit ratio after time 500 in the figure. We can see that, with only 10% of the Interests being malicious (*i.e.*, $\gamma = 10\%$), the normal users' averaged hit ratios degrade from 8.47% to 6.56%. Note that such a reduction leads to considerable traffic on backbone networks. For example, suppose that the clients in PoP_7 request contents at a rate of 100 Gbps. Such a hit ratio reduction suggests that the NDN router in PoP_7 needs to download approximately 2 TB of additional data on a daily basis, which is a non-trivial overhead. From the figure, we also find that the attacker has a very high hit ratio, indicating that he has successfully built false localities for the unpopular objects in the set $\mathcal{P}$.



(a) Damage ratio (b) Attacker's hit ratio (c) Percentage of the polluted cache space

Fig. 3. (a) Damage ratio, (b) attacker's hit ratio, and (c) percentage of the polluted cache space in the NDN router under various γ and θ settings.

Finally, we observe that the overall hit ratio perceived by the router in PoP_7 is even slightly increased after the attack is launched. The observation suggests that, unlike locality-disruption pollution [10] [11], false-locality pollution does not necessarily reduce the overall hit ratio and thus is difficult to detect.

In the next experiment, we explore the attacker's pollution strategy by varying the power ratio γ from 3% to 10% and the range ratio θ from 0.1 to 1.0. We use the *damage ratio* metric, which is defined as $\frac{(hr_n - hr_a)}{hr_n}$, where hr_n and hr_a denote the normal users' hit ratios in the absence/presence of pollution, to measure the pollution effectiveness [6]. Fig. 3(a) plots the damage ratios under various power and range ratio settings. From the figure, we can see that, under a given attacking power constraint, the attacker needs to carefully select his range ratio, *i.e.*, the number of objects in $\mathcal{P}$, to effectively pollute the NDN router. For example, given a 3% power ratio constraint, the attacker achieves the highest damage ratio under a range ratio of $\theta = 0.2$. In other words, the attacker needs to select a set $\mathcal{P}$ of unpopular objects such that $\mathcal{P}$'s cardinality is equivalent to 20% of the router's cache space to achieve the highest pollution effectiveness. The attacker will not obtain a better result by putting more or fewer unpopular objects in $\mathcal{P}$.

We explain the balance in selecting the range ratio θ as follows: If the set $\mathcal{P}$ is too small, even though all the objects in $\mathcal{P}$ have been successfully injected into the NDN router's Content Store, the attacker still cannot occupy excessive cache space, leading to a suboptimal damage ratio. On the other hand, if $\mathcal{P}$ is too large, the attacker cannot request each object in $\mathcal{P}$ with sufficient frequency with limited attacking power and thus will face difficulties in keeping them in the Content Store.

To support this point, in Fig. 3(b) and (c), we plot the attacker's hit ratios and percentages of the NDN router's cache space polluted by the attacker under various power and range ratio settings. The pollution percentage is obtained by calculating the product of the range ratio θ and the attacker's hit ratio. From the two figures, we can see that the highest hit ratio of the attacker does not lead to the highest damage ratio, and the attacker needs to balance the number and hit ratios of the objects in $\mathcal{P}$ to maximize the pollution effectiveness.

TABLE I
STRUCTURAL FEATURES OF FOUR REPRESENTATIVE SPRINT POPS

PoP	New York	Dallas	Chicago	Atlanta
Backbone routers	8	4	7	4
Access routers	100	97	241	67
Router-level links	111	200	393	109
Topology depth	10	8	9	6

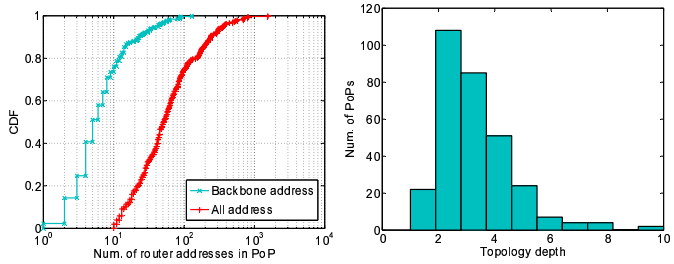
IV. THE NDN POP NETWORK

In our previous study, we assumed that each PoP has only one router, which obviously over-simplifies the network structure in NDN PoPs. Before discussing methodologies against pollution attacks, in this section, we first examine the PoPs in real-world ISP networks and consider the following question: *what will the PoPs look like under NDN?*

We employ the large-scale topology measurement dataset of Rocketfuel [18] to study the real-world PoPs. Usually, there are two types of routers within a PoP, namely, the *backbone router*, which connects to backbone routers of other PoPs over the ISP's long-distant backbone links, and the *access router* that interconnects backbone routers and clients. A PoP is usually characterized by a complex network structure with dozens or even hundreds of routers, and there are substantially more access routers than backbone routers. As an example, in Table I, we list the backbone and access routers, router-level links, as well as the depths of the network topology (which is defined as the longest loop-free path between a backbone router and an edge access router) for four representative Sprint PoPs in Rocketfuel. We can see that all the PoPs have many routers and router-level links; however, there are relatively few backbone routers in the PoP networks.

We also employ a recent Internet topological measurement study of ChinaNet [19] to examine the PoP networks. Fig. 4(a) compares the backbone routers' IP addresses and all router addresses discovered in the 308 ChinaNet PoPs[1], and Fig. 4(b) presents the topology depths of these PoP networks. The figures confirm our observation that 1) a PoP network has a complex internal structure with many routers and router-level links and that 2) there are much fewer backbone routers that

[1]The ChinaNet dataset in [19] does not provide the alias resolution results; therefore, here, we present router IP addresses instead of routers for the PoPs.



(a) (b)

Fig. 4. (a) Backbone addresses and all router addresses and (b) topology depths of the 308 ChinaNet PoPs.

are at higher positions of the network hierarchy compared to access routers.

In light of the real-world PoP networks, we conjecture that a future NDN PoP will also be composed of a few backbone routers and a large number of access routers interconnecting the backbone routers and user clients. Furthermore, we assume that the backbone routers have much larger caching capacities than do the access routers and are responsible for most of the cache hits within the PoP; in addition, some access routers may operate under a "forwarding-only" mode, that is, the routers only forward Interest and Data messages and do not cache any data objects because they are not equipped with Content Stores.

We justify our assumption on NDN PoPs with the following observations: First, various studies [20] [21] report that universal caching (*i.e.*, caching on every router) is far from the optimal cache deployment strategy and should not be automatically assumed. Second, recent studies [22] [23] show that it is beneficial to allocate cache space on nodes that are at the core of the network. Summarizing these observations, we believe that large Content Stores should be deployed on backbone routers, which indeed serve as "hubs" of the PoP network.

Placing high-performance caches on backbone routers also conforms with current PoP network architecture design principle [24]. In current PoP design, backbone routers provide high-performance switching over long-haul backbone networks, whereas access routers achieve relatively low performances but provide other features such as diversity of interfaces for supporting a wide range of access technologies and redundant paths for network reliability. Because the performance benefit of in-network caching in NDN is to avoid transmitting data redundantly over long-distant backbone links, it is reasonable to place the large-volume high-speed Content Stores on backbone routers rather than on access routers.

Note that,in the above-described NDN PoP network, an Interest is unlikely to hit on its first-hop access router and is more likely to traverse a router-level path to hit on a backbone router in the PoP. In the next section, we will exploit the diversity of these paths to detect a pollution attack.

V. Anti-pollution Methodology

In this section, we present our methodology for detecting and mitigating false-locality pollution attacks under NDN. We first present our basic idea and analyze its rationale; then, we describe our proposed anti-pollution algorithm in detail. Finally, we discuss how to implement the algorithm in an NDN backbone router.

A. Basic idea

We exploit the following observation to detect the false-locality pollution attack: A data object that is truly popular will be requested by many normal users at many different locations, and the Interests requesting this object will traverse many different router-level paths between the requesting clients and the backbone routers within the PoP. On the other hand, for an unpopular object that is exploited by an attacker for cache pollution (*i.e.*, an object in $\mathcal{P}$), most of its requests are sent by the compromised hosts under the adversary's control. Although the compromised hosts may issue as many Interest messages as the ones sent out by normal users for a popular object, they are few in number and thus are unlikely to be as widely located in the network as the normal users. Therefore, the malicious Interest messages are unlikely to traverse as many different paths as the messages requesting a truly popular object.

Based on this observation, our basic idea for thwarting a false-locality pollution attack is as follows: For a data object that is cached by a backbone router, we expect to see the Interest messages requesting this object traverse many different router-level paths within the PoP. *If we do not observe such a path diversity, it is very likely that the cached object is not truly popular and is exploited by an adversary for polluting the NDN network; thus, it should be evicted from the Content Store.*

There may be two concerns regarding our basic idea. First, the compromised hosts might be as widely located in the network as normal users. However, we believe that this is unlikely to occur because the chance of an adversary infecting a host is very low. For example, it is reported that Code Red, one of the largest and fastest Internet worms, can infect $360,000$ hosts among 2^{32} IP addresses. The infection chance, which is 8.4×10^{-5}, is extremely low [25]. In other words, even with an infection capacity as high as that of Code Red, an adversary can on average compromise only one host in a network with nearly $12,000$ IP addresses, which is much larger than most PoP networks' address spaces.

The other concern is the geographical locality of the popular objects. Specifically, some popular objects may only be requested by users from one or a few locations and are not popular across the network. Geographical locality is observed between different countries and continents, where users have different languages and culture backgrounds, but is rarely observed within culturally homogeneous regions [26] [27]. For a PoP network that serves a limited geographical region (*e.g.*, a university campus or a residential area), users tend to have highly homogeneous preferences and always prefer the same popular contents. For example, a recent measurement study [27] on YouTube shows that the popular videos requested by users in one residential network are also popular in other residential networks. Because an NDN backbone router only

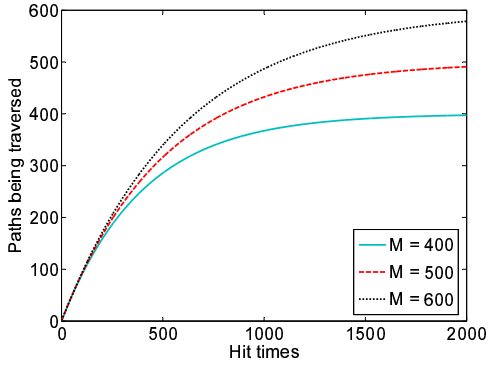


Fig. 5. Average number of distinct paths traversed by n Interest messages given M total paths within a PoP.

caches the very popular data objects, we believe that there is no geographical locality for these objects within a PoP network.

B. Analysis

We present a simple analysis to understand the relationship between a data object's popularity and the diversity of the paths its Interest messages traverse. Suppose that, within a PoP network, there are M distinct paths between the clients and a backbone router and that, for a popular data object, the probability that an Interest message traverses the i^{th} path to reach the backbone router is p_i ($\sum_{i=1}^{M} p_i = 1$). Then, our question is as follows: Given that a cached object o has been hit n times, how many distinct paths should be traversed by the n Interest messages to reach the backbone router?

The problem is a "balls into bins" problem [28]. Consider one specific path, say, path i; the probability that none of the n Interests traverses it is $(1-p_i)^n$. If we use a random variable x_i to denote whether this path has been traversed by any of the n Interests ($x_i = 1$) or not ($x_i = 0$), x_i follows a Bernoulli distribution:

$$\begin{aligned} \Pr(x_i = 0) &= (1-p_i)^n \\ \Pr(x_i = 1) &= 1-(1-p_i)^n \end{aligned}$$

Among all M distinct paths, the n Interest messages traverse X paths, where $X = \sum_{i=1}^{M} x_i$, and it is clear that

$$m = E[X] = M - \sum_{i=1}^{M} (1-p_i)^n \tag{2}$$

Furthermore, for the special case that each path is traversed with equal probability, *i.e.*, $p_i = \frac{1}{M}$, X is indeed a Binomial random variable with distribution

$$\Pr(X = k) = \binom{M}{k} (1-q)^k q^{M-k}$$

where $q = \left(1 - \frac{1}{M}\right)^n$, and we have

$$m = E[X] = M - M\left(1 - \frac{1}{M}\right)^n \tag{3}$$

In Fig. 5, we apply Eq. (3) to calculate the distinct paths being traversed by n Interests and plot the numerical relationship under different M total paths within a PoP. We can see that the more popular an object is, the more distinct paths its Interests traverse.

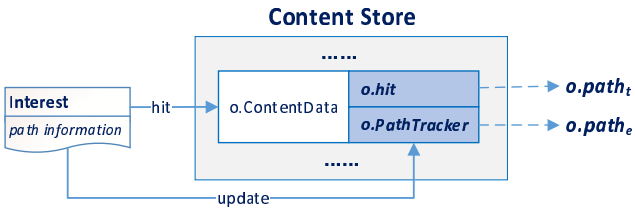


Fig. 6. Demonstration of processing an incoming Interest by a backbone router.

C. The anti-pollution algorithm

After analyzing the basic idea, in this section, we describe our proposed algorithm for thwarting false-locality pollution attacks in detail.

We first augment NDN's Interest message to enable it to record the path it has traversed from the client to the backbone router. Specifically, when an NDN router receives an Interest, before processing it with the Content Store or FIB, the router first appends the address or alias of its interface on which the Interest is received for the message. When all the routers along the Interest traversing path append their interfaces, upon reaching the backbone router, the Interest message should be carrying the path information as an array of the router interfaces along the path. Unlike address-based detection [6], which can be bypassed with forged addresses, it is difficult for an attacker to forge the path information on compromised hosts. Note that path information is generated on routers hop by hop, and the clients are connecting to edge access routers as their first hop. Therefore, the Interest messages received by an edge router from a client should not carry any path information. If an edge access router receives an Interest message with path information from a client-facing interface, it is clear that the path information is forged, and the router should discard the Interest without further process.

With path information piggybacked on Interest messages, we propose to allow the backbone router to trace paths in its Content Store. Specifically, for each cached data object, say, object o, we associate it with a counter $o.hit$, which records how many times o has been hit since the last time o was examined against pollution. We also associate object o with a data structure named *PathTracker*, which traces the number of distinct paths traversed by the Interest messages requesting o within the PoP. We will describe the methodologies for constructing, updating, and querying PathTracker in the next section.

We demonstrate the procedure for processing an incoming Interest message in Fig. 6: When an Interest requesting object o reaches and hits on the backbone router, the router increments $o.hit$ and updates the associated $o.PathTracker$ with the carried path information. When queried, $o.PathTracker$ returns the number of the distinct paths as $o.path_e$. Note that, by knowing $o.hit$ and assuming o to be truly popular, we can also calculate $o.path_t$, the expected number of distinct paths

Algorithm 1 Path update

1: **procedure** UPDATE($Interest(o)$) ▷ $Interest(o)$ is an Interest message requesting object o and hits on the backbone router
2: **if** $Interest(o).path$!= null **then**
3: $o.hit++$;
4: Update $o.PathTracker$ with the path information piggybacked on $Interest(o)$;
5: **end if**
6: Process $Interest(o)$ with Content Store;
7: **end procedure**

Algorithm 2 Anti-pollution algorithm

1: **procedure** DETECT_POLLUTION
2: **for** each cached object o in Content Store **do**
3: **if** $o.hit \geq \eta$ **then**
4: Query $o.path_e$ from $o.PathTracker$ with Eq. (5) or Eq. (6) and calculate $o.path_t$ using Eq. (2);
5: Calculate the ratios of $o.r_t$ and $o.r_e$ with Eq. (4);
6: **if** $o.r_e > \xi \times o.r_t$ **then**
7: Decide that o is exploited for pollution and evict;
8: **else**
9: Clear $o.hit$ and $o.PathTracker$;
10: **end if**
11: **end if**
12: **end for**
13: **end procedure**

traversed by o's Interests, by applying Eq. (2). Algorithm 1 formally describes the procedure for updating path information on an NDN router.

With the queried and calculated distinct paths of $o.path_e$ and $o.path_t$, our pollution detection algorithm is very simple. For each cached object o, if o has been hit more than η times, *i.e.,* $o.hit \geq \eta$, where η ($\eta > 0$) is a threshold parameter, we compare the two ratios between the Interest traversing paths and the hit times using $o.path_t$ and $o.path_e$ as

$$\begin{aligned} r_t &= \frac{o.hit}{o.path_t} \\ r_e &= \frac{o.hit}{o.path_e} \end{aligned} \tag{4}$$

Suppose that o is exploited by a pollution attacker, which means that o is only requested by compromised hosts. The number of paths actually traversed by its Interests $o.path_e$ should be smaller than that of $o.path_t$; therefore, we have $r_e > r_t$. If r_e is sufficiently large such that $r_e > \xi \times r_t$, where ξ ($\xi > 0$) is another threshold, we can decide that o is an exploited object and evict it from the Content Store. Otherwise, o is considered as truly popular, and we clear its associated $o.hit$ and $o.PathTracker$ so that the object will be examined once again after it has accumulated η hits. Algorithm 2 presents the formal description of the anti-pollution algorithm.

One concern for the algorithm is that the path information carried by the Interest messages may expose the internal structure of the PoP network. To protect such information, on cache miss, the border router may choose to remove the path information when forwarding an Interest out of a PoP or AS.

Another concern is the processing and network overheads caused by carrying path information with Interest messages.

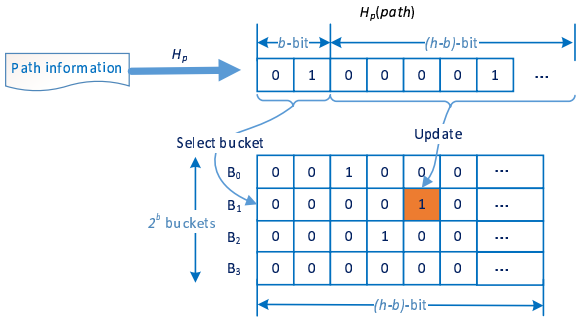


Fig. 7. Demonstration of path update for PCSA-based PathTracker.

We believe that the processing overhead incurred by appending interface information is acceptable for NDN routers because routers on IP networks have been performing similar operations for decades. For example, an IP router must extract, modify, and write back the time-to-live (TTL) field and recalculate the checksum for every IP packet it forwards at line speed. Concerning network overhead, suppose that appending a router interface requires 20 bytes, which is sufficient for router interface aliases [29] because the average router-level hop counts between any two addresses on the Internet is 15 [30]. Even for an Interest message that traverses the entire Internet, the average overhead is only 300 bytes. Compared with a Data message of several megabytes, such overhead is trivial. Furthermore, because the path information may be restricted within a PoP or AS network, which means that the router-level paths will be much shorter, the overhead can be further reduced.

D. PathTracker implementation

Our proposed anti-pollution algorithm requires that each cached data object be associated with a data structure named PathTracker, which traces the distinct paths traversed by the Interest messages for this object within the PoP. In this section, we discuss how to implement such a data structure on an NDN backbone router. We have three requirements for PathTracker: First, it should be memory-inexpensive and not consume an excessive amount of memory space on the router. Second, its operations of updating a path and querying path diversity should be computational inexpensive and easy to implement. Finally, it should be sufficiently accurate to differentiate between a truly popular data object and an object that is exploited as pollution based on their path diversities.

1) Implementation with PCSA: We first consider implementing PathTracker using the probabilistic counting with stochastic averaging (PCSA) technique, which is a technique for estimating cardinalities of large sets [31] [32].

As demonstrated in Fig. 7, a PCSA-based PathTracker maintains bitmaps composed of 2^b buckets $B_0, ..., B_{2^b}$, where each bucket has $(h - b)$ bits initialized as zero. When an Interest carrying its path information hits on the router, we employ a hash function $H_p(\cdot)$ to calculate a h-bit hash value from the path and take the first b bits to decide which bucket to update. The chosen bucket is updated as follows: we count

the longest sequence of '0's in the remaining $(h-b)$ bits of the hash value from the most significant bit. If there are k consecutive '0's, we set the bucket's $(k+1)^{th}$ bit as '1'. Fig. 7 demonstrates an example in which $b=2$ and there are four '0's leading the remaining $(h-b)$-bit hash value. In this case, we select the 2^{nd} bucket B_1 and update $B_1[4]$ as '1'.

To query the distinct paths traced by a PCSA-based PathTracker, for each bucket B_i, we use z_i to denote the position of the most significant '0' in the bucket and let $\bar{z}$ be the average of z_is for all the buckets. According to [32], the number of the distinct paths can be estimated as

$$path_e = 2^b \times 2^{\bar{z}} \times 0.79402 \tag{5}$$

For analyzing the memory overhead, consider an h-bit hash function. We have 2^b buckets, each with $(h-b)$ bits. Therefore, the bitmap consumes $2^b \times (h-b)$ bits. For example, if we employ a 32-bit hash function and use the first $b=3$ bits to select the bucket, the PathTracker consumes 232 bits, which is a trivial overhead compared with a typical data object of several megabytes. The PCSA-based PathTracker is also computational inexpensive, with only one hash calculation being required per update.

2) Implementation with Bloom filter: We also consider implementing PathTracker with a Bloom filter (BF). A BF-based PathTracker is implemented as an array $b[1,...,m]$ of m cells, with each cell being a bit initialized as zero. Unlike the conventional Bloom filter, which employs multiple hash functions for membership tracing, we employ only one hash function $H_b(\cdot)$ to determine the mapping between paths carried by Interest messages and BF cells. More specifically, to update a path, we compute an h-bit hash value $H_b(path)$ and update the cell $b[H_b(path)]$ as '1'.

For querying the distinct paths traced by a BF-based PathTracker, we apply the following formula:

$$path_e = \frac{\ln\left(1-\frac{t}{m}\right)}{\ln\left(1-\frac{1}{m}\right)} \tag{6}$$

where m is the Bloom filter size and t is the number of '1's in the Bloom filter [33].

The BF-based PathTracker is computational inexpensive, with only one hash calculation being required per update. For memory overhead, note that, given the maximum number of items n traced by a Bloom filter, the Bloom filter should have at least $m=\frac{n}{\ln 2}$ cells for minimizing the false positive probability [34]. Thus, a BF-based PathTracker with 232 bits is able to trace up to 160 distinct paths without incurring significant errors, which should be sufficient to differentiate between a truly popular object and an exploited object based on their path diversities.

Both the PCSA-based and BF-based counting techniques have their limitations. PCSA's average error of $1.3/\sqrt{2^b}$ suggests that it would not be sufficiently accurate with a small b, whereas Eq. (5) suggests that, when b is large, this technique will be inaccurate for tracing a small number of items. On the other hand, with a fixed size, a Bloom filter will produce more false positive errors when tracing more items. In Section VI, we will systematically evaluate our proposed anti-pollution algorithm with the two PathTracker implementations.

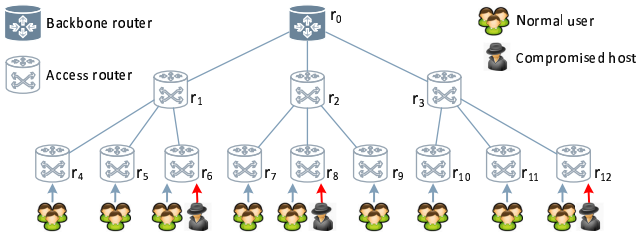


Fig. 8. Demonstration of a k-ary tree PoP network.

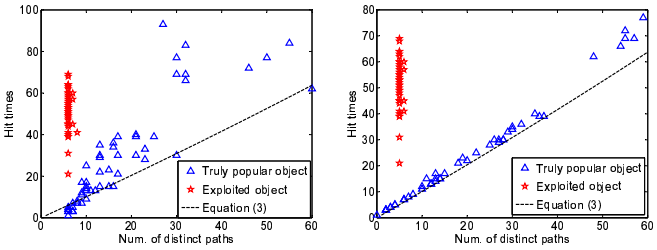
VI. Evaluation

A. Experimental setup

We conduct simulation experiments to evaluate effectiveness of our proposed anti-pollution algorithm. As in Section III, we simulate a content catalog of one million data objects with the MZipf popularity model. We employ the PoP-level topology in Fig. 1 and apply the LCE cache placement strategy and LFU eviction strategy. However, unlike the previous experiments, for each PoP, we simulate its router-level topology as a k-ary tree, where only the root node of the tree is a fully functional NDN backbone router, and all the other nodes are forwarding-only access routers. Specifically, the access router processes Interest and Data messages with its FIB and PIT components but does not cache any data object. Each access router in the k-ary tree has one upward interface and k downward interfaces. A router's upward interface is connected to a downward interface of its parent router, and each of an edge access router's downward interfaces is connected to a *client subnet*. Note that, within our PoP network, each client subnet has a distinct path to reach the backbone router. An exemplary PoP network is presented in Fig. 8. Note that, although it is very simple, the k-ary tree PoP network captures our observations and analysis on real-world ISP PoPs in Section IV.

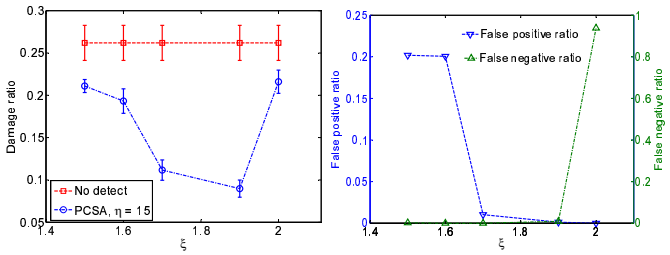
As in today's Internet, we allow NDN backbone and access routers to have aliases on their interfaces. In our simulation, we consult Sprint's router naming convention [29] and denote a backbone router's interface alias in the form of `sl-city-bb-0-n` and an access router's interface alias as `sl-city-gw-m-n`, where `city` is the city code, *e.g.,* "`ny`" or "`la`"; m is the router ID; and n is the index of the router's interface. The path information carried by an Interest message is a concatenation of the interface aliases along the path. For example, in Fig. 8, the path from a subnet connecting router r_7 to the backbone router r_0 could be "`sl-ny-gw-7-2` → `sl-ny-gw-2-1` → `sl-ny-bb-0-3`".

In our simulation, for each PoP, we simulate a k-ary tree PoP network with $k=8$ and depth $d=3$; thus, the network has $k^d=512$ client subnets (and router-level paths) and 73 routers. We assume that truly popular objects are requested by clients in each subnet with equal probability. To trace the path diversity, we employ the PCSA-based or BF-based PathTracker in the backbone router r_0. For the PCSA-based PathTracker, we use a 32-bit hash function $H_p(\cdot)$ and let $b=3$; thus, the data structure consumes 232 bits. For the BF-based PathTracker, we also employ a 232-bit Bloom filter so that



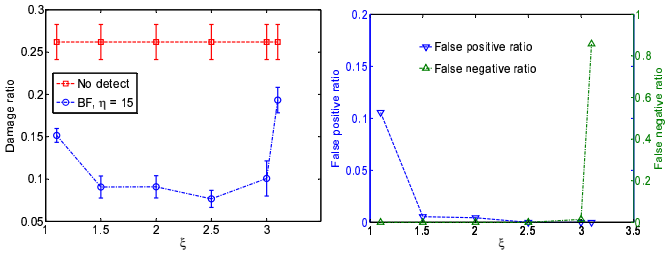
(a) PCSA-based PathTracker (b) BF-based PathTracker

Fig. 9. Path diversities traced by PCSA- and BF-based PathTracker under pollution attack.



(a) Damage ratio (b) Error ratio

Fig. 10. Damage ratios and error ratios of the PCSA-based anti-pollution algorithm with $\eta = 15$ and various ξ values.



(a) Damage ratio (b) Error ratio

Fig. 11. Damage ratios and error ratios of the BF-based anti-pollution algorithm with $\eta = 15$ and various ξ values.

the two implementations have identical memory overhead. In the following, we refer to the anti-pollution algorithm with PCSA-based and BF-based PathTracker implementations as the PCSA-based and BF-based algorithms, respectively.

B. Pollution thwarting

We consider a case wherein an adversary controls a number of compromised hosts in 5 different client subnets in PoP_7 and uses them to launch a pollution attack. The attacker has a power ratio of $\gamma = 10\%$, sets his range ratio as $\theta = 0.6$, and requests a set $\mathcal{P}$ of the least popular objects in the content catalog to pollute the NDN network. From Section III, we know that such an attack is quite effective, with an average damage ratio as high as 0.262.

1) Tracing path diversity: We first examine the path diversities traced by PathTracker. Specifically, we allow the incoming Interests to update the PCSA-based or BF-based PathTracker associated with each cached object by running Algorithm 1; however, we do not execute Algorithm 2 to detect and evict the object replicas that are injected by the attacker.

In Fig. 9(a) and (b), we plot the hit times and the distinct Interest traversing paths traced by the PCSA-based and BF-based PathTracker for all the cached objects at time 300. Each plot in the figure represents a cached object, and we also apply Eq. (3) to compute the expected number of Interest traversing paths and plot the curves on the figures.

We make the following observations from the figures: First, it is feasible to use the ratio between hit times and distinct paths to differentiate a truly popular object from an exploited unpopular object. As one can see in the figures, when a truly popular data object is constantly requested by clients from different subnets, many different paths are traversed by the Interests. In contrast, for an exploited object, because most of its Interest messages are issued by compromised hosts in the 5 subnets, we do not observe such a proportional increase in the number of traversed paths. Our second observation is that, when tracing a few to dozens of paths, the BF-based PathTracker is more accurate than the PCSA-based algorithm because the latter tends to over-estimate the paths when there are few hits and under-estimate them when the cached object has many hits.

2) Mitigating pollution attack: We then apply the anti-pollution algorithm in Algorithm 2 to detect and evict the objects that are exploited by the pollution attacker. To evaluate the effectiveness of the pollution mitigation, we compare the damage ratios on PoP_7's backbone router with/without the algorithm. We also calculate the *false positive* and *false negative error ratios* of the eviction decisions made by the algorithm. The false positive error ratio is defined as

$$r_{fp} = \frac{\text{truly popular objects that are mistakenly evicted}}{\text{all evicted objects}}$$

and the false negative ratio is

$$r_{fn} = \frac{\text{objects in } \mathcal{P} \text{ that are missed by the algorithm}}{\text{all the objects in } \mathcal{P} \text{ examined by the algorithm}}$$

Recall that our proposed algorithm has two thresholds, η and ξ, and a cached object o will be considered as exploited only when $o.hit \geq \eta$ and $o.r_e > \xi \times o.r_t$. Note that the smaller the threshold η is, the more frequently the algorithm examines the cached objects. However, when η is too small, it would be difficult to find a feasible value of ξ to accurately differentiate the truly popular objects from the exploited objects because there are few samples. Therefore, in the following experiments, we first select a value for the threshold η; then, we vary ξ to explore the feasible parameter settings. We refer to a setting of η and ξ as feasible only when both error ratios of r_{fp} and r_{fn} are close to zero, and under a feasible setting, the algorithm should achieve a good performance in mitigating the pollution.

We first employ the PCSA-based PathTracker and execute our proposed anti-pollution algorithm with various η and ξ settings. The results are presented in Fig. 10. In Fig. 10(a), we let $\eta = 15$ and plot the damage ratios under different ξ values. We also plot the damage ratio without the anti-pollution algorithm (the "no detect" curve) for comparison. In Fig. 10(b), we present the false positive and false negative ratios of the algorithm under various ξ values. From Fig. 10(a), we can see that our proposed algorithm with PCSA-based

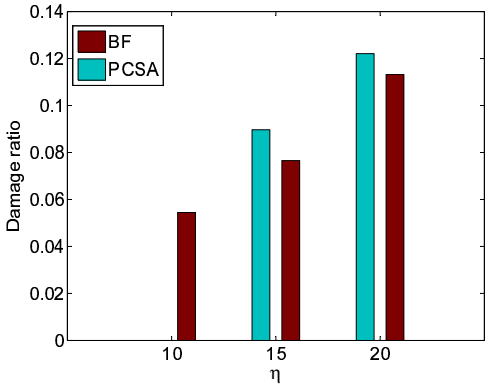


Fig. 12. The lowest damage ratios achieved by the PCSA- and BF-based algorithms under different η settings.

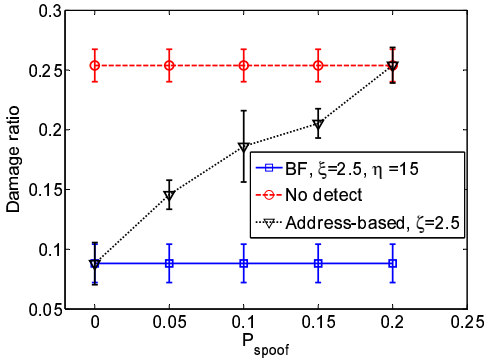


Fig. 13. Damage ratios when applying the BF-based algorithm and address-based method and no pollution detection under address spoofing.

PathTracker can effectively detect and mitigate the pollution; a damage ratio as low as 0.089 can be achieved compared with the damage ratio of 0.262 obtained without the anti-pollution algorithm. Furthermore, by comparing Fig. 10(a) and (b), we find that, to achieve the best performance, the value of ξ must be selected carefully: if ξ is too large, the anti-pollution algorithm makes false negative errors by missing the objects in the set $\mathcal{P}$; on the other hand, when ξ is too small, the algorithm will mistakenly evict some objects that are truly popular due to the estimation errors introduced by the PCSA-based PathTracker.

In Fig. 11(a) and (b), we plot the damage ratios and the detection error ratios of the BF-based anti-pollution algorithm. We also set $\eta = 15$ and vary ξ to search for the best performance. We can see that the BF-based anti-pollution algorithm is also effective according to the resulting very low damage ratios. Moreover, by comparing Fig. 11 with Fig. 10, we find that, with the BF-based PathTracker, the anti-pollution algorithm is more robust in a wider range of feasible ξ than is the PCSA-based algorithm. Such robustness can be explained with the improved accuracy of the BF-based PathTracker over the PCSA-based algorithm, as observed in Fig. 9.

Finally, we directly compare the PCSA-based and BF-based anti-pollution algorithms in Fig. 12. In the experiment, we let $\eta = 10$, 15, and 20 when running the PCSA- and BF-based algorithms. For each η, we explore different ξ values but only plot the lowest damage ratio that can be achieved by the algorithm. Note that, under $\eta = 10$, we are unable to find a feasible ξ for the PCSA-based algorithm due to the inaccuracies of the PCSA-based PathTracker. From Fig. 12, we can see that, for both algorithms, when η increases, the achieved damage ratios increase because the algorithms examine the cached objects less frequently. In addition, we find that, under equal η values, the damage ratios achieved by the PCSA-based algorithm are always slightly higher than the BF-based ratios. This is because, even under the optimal parameter setting, the PCSA-based algorithm still makes a few false positive/negative errors, whereas the BF-based PathTracker does not make errors under the same circumstance.

In summary, we draw the following conclusions:

- Both the PCSA-based and BF-based anti-pollution algorithms are effective in thwarting the false-locality pollution attack.
- The BF-based algorithm is more robust and achieves a better performance than the PCSA-based algorithm thanks to the higher accuracy of the BF-based PathTracker.

C. Resilience against address spoofing

In our proposed anti-pollution algorithm, we exploit the Interest traversing paths for detecting pollution attacks. As analyzed in Section V-C, unlike IP addresses, the path information is appended to the Interest by routers hop by hop along the path; thus, it is very difficult for an attacker to forge compromised hosts.

In this section, we compare our proposed anti-pollution algorithm with the address-based pollution detection method in [6]. Although [6] addresses a pollution attack against file cache systems in IP networks, its idea can by applied to NDN networks. In the address-based method, the cache system monitors the data objects that have high hit ratios and records the source IP addresses of the clients requesting these objects. When an object is observed to be requested by only a small set of IPs or when the ratio between the number of unique IPs and the number of requests for this object is lower than a threshold ζ, the object is considered as injected by an attacker and becomes evicted from the cache.

We apply the same settings as in Section VI-A to evaluate the address-based method. We assume that each client subnet contains 200 clients with different IP addresses. The clients request the data objects according to the MZip model, and the attacker controls compromised hosts in 5 subnets. However, unlike [6], we assume that the attacker is capable of forging the source IP address on a compromised host with a probability P_{spoof}. We evaluate the address-based approach under varying P_{spoof} and compare with our proposed anti-pollution algorithm (with the BF-based PathTracker). Fig. 13 presents the comparison results. From the figure, we can see that our proposed algorithm can always detect a pollution attack regardless of the address spoofing; however, under the address-based approach, the higher the spoofing probability is, the less effective the algorithm is in detecting the pollution. The

TABLE II
HASH CALCULATIONS IN NDN ROUTER COMPONENTS

	FIB	PIT	Content Store	PathTracker
Hash per Interest	2.17	4	1	1

TABLE III
COMPUTATION OVERHEADS AND DAMAGE RATIOS OF NDN BACKBONE ROUTER WITH/WITHOUT PATHTRACKER UNDER NO POLLUTION AND HEAVILY POLLUTED SCENARIOS

	No pollution	Heavily polluted	
	Hash per Interest	Damage ratio	Hash per Interest
w. PathTracker	6.759	0.077	6.705
w/o. PathTracker	6.670	0.255	6.431

observation suggests that, unlike the address-based method, our proposed algorithm is resilient against address spoofing.

D. Computation overhead on backbone router

Our previous analysis of PathTracker shows that it is computational inexpensive when introducing only one hash calculation per Interest message. In this section, we experimentally evaluate the computation overheads of the NDN backbone router with and without the PathTracker component. Note that how to implement an NDN router remains an open problem [35]; therefore, in our evaluation, we refer to the recent progress on NDN router implementations. More specifically, we assume that FIB is implemented as in [36], PIT is implemented as in [37], and the Content Store is implemented as in [35]. In all these components, the only expensive operations are the hash calculations on the Bloom filter and/or the hash table data structures. In Table II, we list the hash calculations per Interest message for each component. Note that, for FIB, the number of required hash calculations is derived from the empirical length distribution of the name prefixes [36], and PIT is implemented as a d-left hash table with $d = 4$ [37].

We examine an NDN backbone router's computation overhead under two scenarios: 1) the PoP network is heavily polluted by an attacker controlling compromised hosts in 5 different client subnets, with the attacker's strategy parameters set as $\gamma = 10\%$ and $\theta = 0.6$, and 2) the PoP network has not been attacked. Table III lists the number of hash calculations per Interest message on the NDN backbone router with and without PathTracker. From the table, we can see that, under both scenarios, incorporating PathTracker only slightly increases the router's overhead but significantly reduces the damage ratio when the network is under attack. The observation is easy to understand because PathTracker incurs significantly fewer hash calculations than do FIB and PIT; furthermore, the PathTracker component is only updated when the Interest hits the Content Store.

E. Attacker's strategies against pollution detection

In this section, we consider the strategies that a false-locality pollution attacker may employ against the our proposed anti-pollution algorithm.

1) Requesting moderately popular objects: One strategy an adversary may employ is to request moderately popular data objects instead of unpopular objects. The moderately popular objects are those that are not sufficiently popular to be cached by the NDN router but remain requested by normal users. By requesting moderately popular objects, the malicious Interest messages are mixed up with the Interests from normal users and thus are more difficult to detect.

We first examine the pollution effectiveness by requesting moderately popular objects. We allow the attacker to control compromised hosts in 5 subnets in PoP_7, set his range ratio θ as 0.6, and vary his power ratio γ from 3% to 10%. Fig. 14 presents the damage ratios when $\mathcal{P}$ is composed of the least popular and moderately popular objects. Note that, under the two cases, the accumulative popularity ρ differs significantly. From the figure, we can see that, comparing with unpopular objects, requesting moderately popular contents leads to a significant reduction in the damage ratio because the objects injected by the attacker into the Content Store are also requested by normal users.

In Fig. 15, we set the attacker's power ratio as $\gamma = 10\%$ and employ the BF-based PathTracker to trace path diversities of the truly popular objects and the exploited objects in the Content Store at time 300. Comparing Fig. 15 with Fig. 9(b), we can see that, by requesting the moderately popular objects, the attacker can substantially increase the path diversities for the objects in $\mathcal{P}$; for example, although the compromised hosts exist in only 5 subnets, the BF-based PathTracker has traced as many as 17 different paths traversed by the Interest messages requesting an exploited object. The increased path diversity is not difficult to understand because, in addition to the compromised hosts, the exploited objects are also requested by normal users from other subnets within the PoP.

Finally, we employ the BF-based anti-pollution algorithm to detect the attack. To address the increased path diversities of the malicious Interests, we set the threshold parameter η as 25 to allow the PathTracker to trace more paths before making the eviction decisions. Fig. 16 presents the damage ratios achieved under various ξ values, and we also plot the damage ratio without the pollution detection in the figure. Note that when ξ is too small, the false positive errors made by the algorithm (i.e., mistakenly evicting truly popular objects) lead to a damage ratio that is even higher than without the pollution detection. Comparing Fig. 16 with Fig. 11(a), we can see that the anti-pollution algorithm can reduce the damage ratio to as low as the case in which unpopular objects are requested; in other words, the strategy of requesting moderate popular objects does not work against our proposed anti-pollution algorithm. We explain this observation with the fact that, although the exploited objects are cached in the Content Store longer under a higher threshold η, they are less "harmful" in polluting the NDN network than are the unpopular objects.

2) Compromising more subnets: The adversary may employ a more direct way of increasing the path diversities of the exploited data objects by compromising hosts from more client subnets. In this section, we consider the cases that the attacker is capable to control compromised hosts in 15 and 25 subnets. The attacker applies the same pollution strategy of

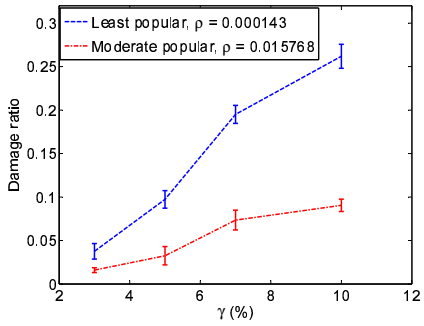


Fig. 14. Damage ratios when an attacker requests the least popular and moderately popular data objects with various power ratios.

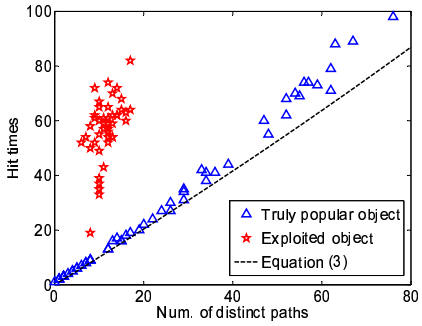


Fig. 15. Path diversities of the truly popular objects and the moderately popular objects that are exploited by the attacker.

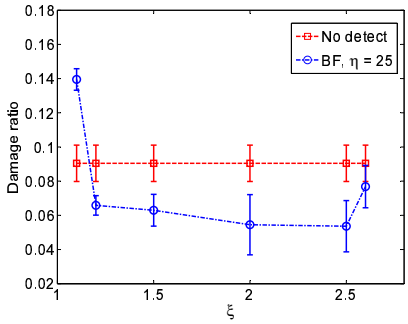
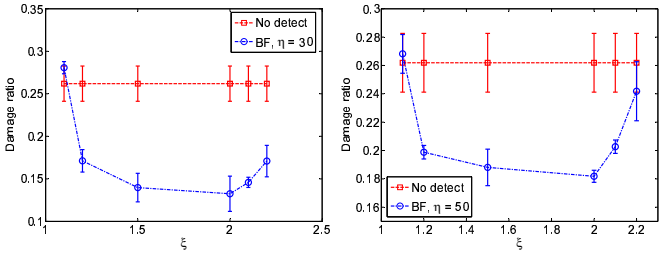


Fig. 16. Damage ratios of the BF-based anti-pollution algorithm with $\eta = 25$ and various ξ values.



(a) Compromising 15 subnets (b) Compromising 25 subnets

Fig. 17. Damage ratios of the BF-based anti-pollution algorithm when there are (a) 15 and (b) 25 compromised subnets in the PoP network.

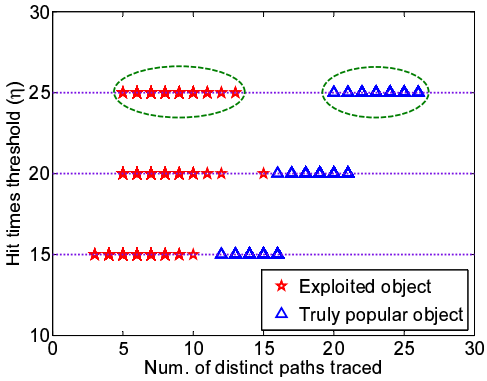


Fig. 18. Inferring the algorithm parameters.

$\gamma = 10\%$ and $\theta = 0.6$ and requests the least popular objects in the content catalog as in the previous experiments.

We employ the BF-based anti-pollution algorithm and set η as 30 and 50 for the cases in which the malicious Interests are from 15 and 25 subnets, respectively. Fig. 17(a) and (b) present the damage ratios obtained by the algorithm under the above two cases, respectively. We can see that the BF-based algorithm still can significant reduce the damage ratios. By studying Fig. 17 and Fig. 12(c), we find that, with a greater number of compromised subnets, our anti-pollution algorithm becomes less effective because we must employ a larger η to address the increased path diversities of the malicious Interests, causing the algorithm to examine the cached objects less frequently.

Our observation suggests that, if an attacker can compromise hosts in many client subnets within a PoP, he can indeed improve the resilience of the attack against our anti-pollution algorithm. However, we believe such a strategy is impractical in practice. Recall that the chance of infecting one host by Code Red, one of the largest and fastest Internet worms, is only 8.4×10^{-5} [25]. If we assume that each client subnet has a $/24$ IP address space, the probability of compromising any host in a subnet is as low as 2.15×10^{-2}; therefore, it is almost infeasible for an adversary to compromise many subnets within a PoP simultaneously.

In summary, we conclude the following:

- The strategy of requesting moderately popular objects is not effective against our anti-pollution algorithm because the moderately popular objects are less harmful than the unpopular ones when polluting an NDN network.
- A pollution attacker can improve the resilience of their attack against our anti-pollution algorithm by compromising more subnets; however, such a strategy is impractical because it is very difficult for the attacker to simultaneously compromise many subnets within a PoP network in practice.

F. Discussion

Finally, we discuss how to determine the two threshold parameters of η and ξ for our algorithm. In general, the parameter depends on the path diversities of the malicious Interests sent by the pollution attacker, which further depend on the attacker's strategy, *i.e.*, how many hosts and subnets are compromised, how frequently to request an exploited object, *etc.*. Usually, such information is unavailable to network administrators.

To overcome this problem, we suggest to choose a few η thresholds arbitrarily and employ the BF-based PathTracker to trace the path diversities of the cached objects under these thresholds. When there exists a pollution attack and η is sufficiently large, the two types of data objects are expected to exhibit different degrees of path diversities. For example, Fig. 18 presents the path diversities for the cases of $\eta = 15$, 20, and 25 under the moderate-popularity pollution attack. When $\eta = 25$, we observe two clusters formed by the two types of data objects far apart on the graph and can easily determine the feasible η and ξ values. Furthermore, clustering machine

learning algorithms, such as $k-$means, can be employed as an offline algorithm to make the procedure more automated.

VII. Conclusion

In this paper, we focus on the threat of false-locality pollution attacks in NDN networks. We present a unified model to profile an attacker's strategy in polluting an NDN network and show that, with limited resources, a pollution attacker can cause considerable damage to the NDN network. Based on our analysis on real-world PoP networks, we present an anti-pollution algorithm that exploits the diversity of the Interest traversing paths within an NDN PoP and propose two methodologies based on the probabilistic counting and Bloom filter techniques to implement the algorithm on NDN backbone routers. Our proposed algorithm is lightweight in terms of the memory and computation overheads and is resilient against forged Interest messages. Simulation experiments suggest that our proposed anti-pollution algorithm is effective in thwarting a false-locality pollution attack, and the strategies that an adversary may take against our algorithm are either ineffective or impractical in the real world.

Future work includes implementing the anti-pollution algorithm on the NDN testbed [38] based on protocol oblivious forwarding (POF) [39] SDN technology and evaluating the algorithm in real-world deployments.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 61202405), the sub task of Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA06011201), and the Anhui Provincial Natural Science Foundation (Grant No. 1608085MF126).

References

[1] G. Xylomenos, C. Ververidis, and et al., "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2013.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. of CoNEXT'09*, Rome, Italy, Dec. 2009.

[3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, and et al., "Named data networking," *ACM SICOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[4] F. Guo, J. Chen, and T. cker Chiueh, "Spoof detection for preventing DoS attacks against DNS servers," in *Proc. of ICDCS'06*, Jul., Lisboa, Lisboa 2006.

[5] J. Liang, R. Kumar, Y. Xi, and K. W. Ross, "Pollution in P2P file sharing systems," in *Proc. of IEEE INFOCOM'05, 2005*, Miami, FL, USA, Mar. 2005.

[6] L. Deng, Y. Gao, Y. Chen, and A. Kuzmanovic, "Pollution attacks and defenses for Internet caching systems," *Computer Networks*, vol. 52, no. 5, pp. 935–956, 2008.

[7] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "Dos & ddos in named data networking," in *Proc. of ICCCN'13*, Nassau, Bahamas, Aug. 2013.

[8] C. Ghali, G. Tsudik, and E. Uzun, "Network-layer trust in named-data networking," *ACM SICOMM Computer Communication Review*, vol. 44, no. 5, pp. 13–19, 2014.

[9] Q. Li, X. Zhang, Q. Zheng, R. Sandhu, and X. Fu, "LIVE: Lightweight integrity verification and content access control for named data networking," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 308–320, 2015.

[10] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in *Proc. of IEEE INFOCOM'12*, Orlando, FL, USA, Mar. 2012.

[11] M. Conti, P. Gasti, and M. Teoli, "A lightweight mechanism for detection of cache pollution attacks in Named Data Networking," *Computer Networks*, vol. 57, no. 16, pp. 3178–3191, 2013.

[12] W. So, A. Narayanan, and D. Oran, "Named data networking on a router: Fast and DoS-resistant forwarding with hash tables," in *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'13)*, San Jose, CA, USA, May 2013.

[13] M. Virgilio, G. Marchetto, and R. Sisto, "PIT overload analysis in content centric networks," in *Proc. of ACM SIGCOMM Workshop on Information-Centric Networking (ICN'13)*, San Jose, CA, USA, Aug. 2013.

[14] G. Mauri, R. Raspadori, M. Gerla, and G. Verticale, "Exploiting information centric networking to build an attacker-controlled content delivery network," in *Proc. of Mediterranean Ad Hoc Networking Workshop*, Vilamoura, Portugal, Jun. 2015.

[15] A. Karami and M. Guerrero-Zapata, "An ANFIS-based cache replacement method for mitigating cache pollution attacks in Named Data Networking," *Computer Networks*, vol. 80, pp. 51–65, 2015.

[16] K. Ali and M. Scarr, "Robust methodologies for modeling web click distributions," in *Proc. of WWW'07*, Banff, Alberta, Canada, May 2007.

[17] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE/ACM Trans. on Networking*, vol. 16, no. 6, pp. 1447–1460, 2008.

[18] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Trans. on Networking*, vol. 12, no. 1, pp. 2–16, 2004.

[19] Y. Tian, R. Dey, Y. Liu, and K. W. Ross, "Topology mapping and geolocating for China's Internet," *IEEE Trans. on Parallel and Distributed Systems*, vol. 24, no. 9, pp. 1908–1917, 2013.

[20] S. K. Fayazbakhsh, Y. Lin, and et al., "Less pain, most of the gain: incrementally deployable ICN," in *Proc. of ACM SIGCOMM'13*, Hong Kong, China, Aug. 2013.

[21] Y. Wang, "Caching, routing and congestion control in a future Information-Centric Internet," Ph.D. dissertation, North Carolina State University, Raleigh, NC, USA, Nov. 2013.

[22] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in *Proc. of IEEE INFOCOM Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN'12)*, Orlando, FL, USA, Mar. 2012.

[23] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for content-centric networking," in *Proc. of ICNP'13*, Gottingen, Germany, Oct. 2013.

[24] "Internet service provider networks: Simplifying pop architectures," http://www.force10networks.com/whitepapers/pdf/App_service_providers.pdf, last accessed 8 Jan. 2016.

[25] C. C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and early warning for internet worms," in *Proc. of ACM Conference on Computer and Communication Security (CCS'03)*, Washington DC, USA, Oct. 2003.

[26] Q. Huang, K. Birman, and et al., "An analysis of facebook photo caching," in *Proc. of SOSP'13*, Farmington, PA, USA, Nov. 2013.

[27] S. Traverso, M. Ahmed, and et al., "Unravelling the impact of temporal and geographical locality in content caching systems," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1839–1854, 2015.

[28] M. Raab and A. Steger, "'Balls into Bins' - a simple and tight analysis," in *Proc. of Second International Workshop on Randomization and Approximation Techniques in Computer Science*, 1998.

[29] "Sprintlink naming conventions," https://www.sprint.net/index.php?p=faq_namingconvention, last accessed 8 Jan. 2016.

[30] E. Katz-Bassett, H. V. Madhyastha, and et al., "Reverse traceroute," in *Proc. of USENIX NSDI'10*, Boston, MA, USA, Mar. 2010.

[31] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *Journal of Computer and System Sciences*, vol. 31, no. 2, pp. 182–209, 1985.

[32] M. Durand and P. Flajolet, "Loglog counting of large cardinalities," in *Proc. of European Symposium on Algorithms (ESA'03), LNCS 2832*, Budapest, Hungary, Sep. 2003.

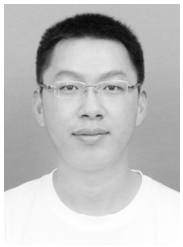
[33] O. Papapetrou, W. Siberski, and W. Nejdl, "Cardinality estimation and dynamic length adaptation for Bloom filters," *Distributed and Parallel Databases*, vol. 28, no. 2, pp. 119–156, 2010.

[34] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of Bloom filters for distributed systems," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, pp. 131–155, 2012.

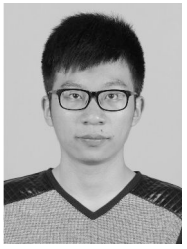
[35] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proc. of ACM SIGCOMM Workshop on Information-Centric Networking (ICN'11)*, Toronto, Ontario, Canada, Aug. 2011.

[36] Y. Wang, B. Xu, and et al., "Fast name lookup for named data networking," in *Proc. of IEEE/ACM IWQoS'14*, Hong Kong, China, May 2014.
[37] H. Yuan and P. Crowley, "Scalable pending interest table design: From principles to practice," in *Proc. of IEEE INFOCOM'14*, Toronto, Canada, Apr. 2014.
[38] Z. Wang, L. Wang, and et al., "An architecture of content-centric networking over protocol-oblivious forwarding," in *Proc. of IEEE Globecom'15*, San Diego, CA, USA, Dec. 2015.
[39] H. Song, "Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane," in *Proc. of SIGCOMM HotSDN Workshop (HotSDN'13)*, Hong Kong, China, Aug. 2013.

Haoran Guo received the bachelor's degree in computer science from the Southwest Jiaotong University in June 2014. He is a master candidate in the School of Computer Science and Technology, University of Science and Technology of China. He is working on the performance and security issues of the future Internet architectures and systems. He is a student member of the IEEE.



Xiaodong Wang received the bachelor's degree in computer science from the Nanjing University of Aeronautics and Astronautics in June 2015. He is currently pursuing the master's degree in the School of Computer Science and Technology, University of Science and Technology of China. His research focus is the future Internet architecture.



Kun Chang received the bachelor's degree in computer science from the Hunan University in June 2015. He is a master candidate in the School of Computer Science and Technology, University of Science and Technology of China. His research interest is focused on the named data networking future Internet.

Ye Tian received the bachelor's degree in electronic engineering and the master's degree in computer science from the University of Science and Technology of China (USTC), in July 2001 and 2004, respectively. He received the PhD degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong in December 2007. He is an associate professor in the School of Computer Science and Technology, USTC. He joined USTC in August 2008. His research interests include future Internet, Internet measurement, software-defined networking, and peer-to-peer networks. He is a member of the IEEE and ACM, and a senior member of the China Computer Federation. He is currently serving as a young associated editor for *Springer Frontiers of Computer Science Journal*.