Towards Persistent Detection of DDoS Attacks in NDN: A Sketch-Based Approach

Zhiwei Xu, *Member, IEEE*, Xin Wang, *Member, IEEE*, and Yujun Zhang, *Member, IEEE*

Abstract—As a promising architectural design for future Internet, Named Data Networking (NDN) relies on data names, instead of destination IP addresses, to deliver data. NDN supports data authenticity and integrity by making public key signatures mandatory on data content and data names. This handles the primary security concern in NDN, but is still vulnerable to new DDoS attacks, including Cache Pollution attacks and Interest Flooding attacks, which degrade NDN transmission significantly, by violating the crucial components of NDN routers. To defend against DDoS attacks in NDN, the most effective way is to persistently detect the malicious traffic and then throttle them. Except for the usual concern of the accuracy and efficiency in attack detection, since these attacks themselves have already imposed a huge burden on victims, to avoid exhausting the remaining resources on the victims for detection purpose, a lightweight detection solution is highly desired. We study DDoS attacks and propose a persistent detection solution based on an observed malicious traffic pattern, which leverages a novel sketch to monitor the malicious traffic in a timely and lightweight way. Additionally, our analysis and experiments demonstrate that, with fixed low resource consumption, the proposed solution can persistently detect DDoS attacks in NDN.

Index Terms—DDoS attacks, persistent attack detection, named date networking, malicious traffic pattern, lightweight, advanced FM sketch

1 Introduction

During the past several decades, Internet has evolved from a simple network for point-to-point communications to a sophisticated global system that supports communication needs of all types of services, including data delivery for information-intensive business, new AI applications, and getting everything connected to enable Intelligent Internet of Things [1]. Existing Internet architecture relies on IP address to guide packet transmissions. However, originally designed to suit the need of connection driven end-to-end communications, this transmission infrastructure is inefficient in supporting data oriented applications. As a promising architectural design for the future Internet, NDN [2], [3] applies hierarchical name-based data transmission to bridge the gap between the current Internet architecture and the new information-intensive applications. *It* leverages in-network caching and multi-path communications to implement efficient hierarchical name-based data retrieval.

In recent years, distributed denial of service (DDoS) attacks have become more common and notorious in Internet. NDN provides a built-in resilience to conventional DDoS attacks. In NDN, to retrieve data, a user needs to send a request (Interest) packet to the data provider, where the Interest packet shares the same hierarchical name with the requested data. After receiving the Interest, a router searches its Forwarding Information Base (FIB) to longest match the name contained in the Interest to find the egress interfaces, through which it can forward the Interests to the down-stream routers. The name and egress interfaces will be stored in the Pending Interest Table (PIT) of this router. The requested data , once found, can be sent back along the reverse path of the Interest packet, and all routers on the path will cache this data to speed up the responses to the subsequent requests on the same data. In this data requesting process, routers on the path aggregate Interest packets requesting for the same data into a PIT entry, and discard the duplicated Interests. Although sending excessive data requests is the most convenient and popular way to mount a conventional DDoS attack in Internet, this aggregation process prevents the sending of a large number of Interest packets to request the same data to deplete the network resources. Thus, conventional DDoS attacks by sending a huge number of duplicated packets can be mitigated in NDN [3]. Unfortunately, NDN is still vulnerable to some new DDoS attacks [4], [5] that maliciously exploit its hierarchical name-based transmission.

More specifically, the two key enablers of highly efficient Data transmission in NDN, Pending Interest Table (PIT) and Content Store (CS), have been proven vulnerable to two respective types of DDoS attacks, i.e., Interest Flooding Attack (IFA) [4] and Cache Pollution Attack (CPA) [6]. Upon the attacks, the PIT or CS on the targeted NDN router could be overwhelmed and thus the corresponding router fails to forward the subsequent Interest packets or satisfy the received Interests with the data cached in the local cache

- *Zhiwei Xu and Yujun Zhang are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China. E-mail: {xuzhiwei2001, zhmj}@ict.ac.cn.*
- *Xin Wang is with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794 USA. E-mail: x.wang@stonybrook.edu.*

Manuscript received 3 March 2021; revised 13 July 2022; accepted 22 July 2022. Date of publication 3 August 2022; date of current version 11 July 2023. This work was supported in part by the National Science Foundation of China under Grants 61962045, 61862048, 61902382, and 61972381, in part by the National Key Research and Development Program of China under Grant 2018YFB1800403, in part by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDC02030500, and in part by the Visiting Scholar Project of China Scholarship Council under Grant 201908150030. (Corresponding autor: Zhiwei Xu.)
Digital Object Identifier no. 10.1109/TDSC.2022.3196187

(CS). Motivated by the importance of addressing security in the early stages of a new Internet architecture, we focus on persistently detecting DDoS attacks over NDN, including both Interest Flooding attack and Cache Pollution attack.

To defend against DDoS attacks in NDN, countermeasures are proposed in literature. For Interest Flooding attacks, Token Bucket [7] and Resource Allocation [8] attempt to proactively balance the PIT usage for interests received by different Interfaces. To ensure the PIT space is available, a predefined PIT usage threshold is introduced [9] to trigger dummy content packet response and release the occupied PIT space. However, lacking of effective attack detection, all interests over the limits will be throttled by these coarse-grained mechanisms. Similarly, the first proactive countermeasure against CPA attacks in NDN [6] compromises the regular cache usage. To throttle the attack traffic more accurately, multiple approaches are proposed to detect CPA or IFA attacks [10], [11], [12]. In [13], the diversity of the Interest traversing paths within an ISP's point-of-presence network is exploited to mitigate the threat of the thwarting pollution attacks. However, these DDoS countermeasures suffer from the high complexity when monitoring different data flows continuously. Built on heavyweight primitives like artificial neural network, approaches in [11], [14], [15], [16], [17] are too complex to run persistently in a realistic NDN router. The long-term excessive consumption of the router resources will cripple the regular data transmission through the router.

In [18], we perform a preliminary study on the CPA detection. In detail, we analyze the malicious traffic pattern of CPA attacks, and propose a light-weight sketch to monitor the malicious traffic pattern based on each individual name prefix. However, since the name prefixes of the requested data vary and have a large number, allocating a sketch for every name prefix will still consume a considerably amount of resources. A practical attack detection solution should be lightweight to reduce the resources needed for running the detection from the victim, which is already overburdened by the attack. Meanwhile, to maintain a high detection performance, the patterns of attack traffic should be correctly and timely identified, that the attack traffic can be accurately throttled without blocking any regular data transmissions. To achieve a persistent detection for all types of DDoS attacks in NDN, we propose to address various challenges for DDoS atttack detection in this paper.

Challenges. In summary, to persistently detect DDoS attacks at high accuracy in NDN, we face multiple challenges:

1) The compromised network nodes may mimic regular packets, thus the malicious packets are indistinguishable from regular ones. To identify attacks, we need to analyze these attacks to provide a rational guarantee for attack detection.
2) The compromised network nodes usually send a considerable number of malicious Interest packets in a short period to deplete Pending Interest Tables or Content Stores. It is a challenge to timely identify the attack traffic before the victim is overwhelmed.
3) To timely discover an attack, monitoring all types of data flows continuously for identifying malicious traffic will lay a huge burden on routers. The challenge is to alleviate this burden without degrading the detection accuracy.

To conquer the above challenges, we first perform a preliminary studies on the impact of DDoS attacks in NDN. We observe that, in order to overwhelm a victim node, an adversary needs to forge a large number of malicious Interest packets that possess a common name prefix, so that these packets can converge at the same network node by following the forwarding principle of longest name prefix match in NDN. To reduce the resource consumption in attack detection, we propose to exploit a novel lightweight and efficient sketch to persistently monitor all types of data flows. The analysis and experiments demonstrate that our approach can largely reduce the traffic monitoring cost while timely and accurately discovering all the aforementioned DDoS attacks.

Contributions. The primary contributions are summarized as follows.

1) We study attacks in NDN and discover an important traffic pattern, based on which we design an solution that can work across-the-board to detect all the major types of DDoS attacks.
2) We propose a lightweight and efficient Flajolet-Martin sketch, LiEffi-FM sketch, that can continuously monitor the pattern of malicious traffics with very low resource consumption.
3) We design a persistent DDoS detection scheme to timely detect DDoS attacks in NDN with high accuracy and low resource consumption.
4) We analyze the overhead and security of the proposed detection solution, and evaluate its performance in a real-world NDN test-bed network [19].

In Section 2, we provide thorough background knowledge on NDN and introduce the primitives we use in our design. We study DDoS attacks in NDN and then present our threat model in Section 3. Following the guidelines proposed in Section 4, we design a light-weight and efficient FM sketch to support continue Interest traffic monitoring in Section 5, and propose the whole attack detection solution to detect all types of DDoS attacks in Section 6. We analyze its overhead and security in Section 7, and evaluate the performance of our design in Section 8. Finally, we outline the related work in Section 9 and conclude in Section 10.

2 Preliminary

2.1 NDN Overview

Unlike IP-based architecture, Named Data Networking (NDN) uses hierarchically structured names rather than IP addresses to guide packet routing and forwarding. The hierarchical structure of IP addresses enables the aggregation which is essential in scaling today's routing system. Similarly, the hierarchical structure of data names used in NDN facilitates the merge of data requests Interests with the same prefix. Data providers announce the name prefixes that cover the data they are willing to serve. These announcements are propagated through the network via a routing protocol, such as OSPF and BGP. Ultimately, according to the same name prefix of Interest packets longest matched

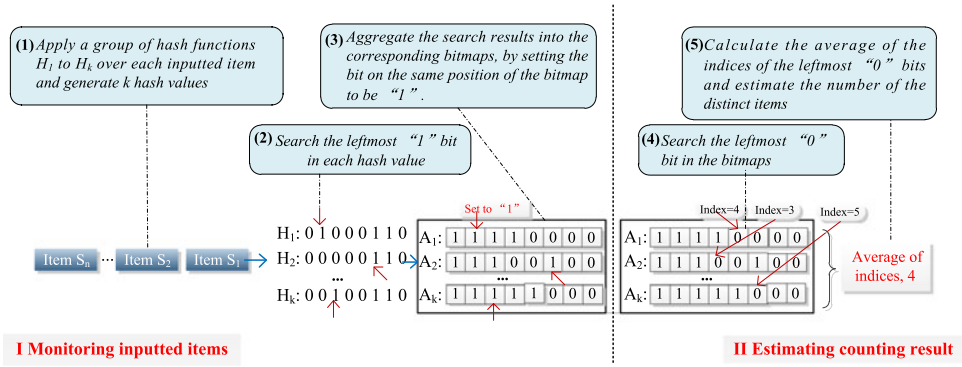


Fig. 1. Basic FM sketch.

by FIBs, NDN routers forward these Interests to the same data providers.

An NDN user (data consumer) retrieves name-based content (data) by three steps: (1) To request name-based data, a consumer (user) first sends an Interest (data request packet) identified by the name of the requested data. (2) After receiving the packet, a router records this Interest and its ingress interface into its Pending Interest Table (PIT). Duplicate Interests are aggregated to one entry by appending the ingress interface list. If the requested data has been cached in its Content Store (CS), the router will respond the Interest with the cached copy. Otherwise, it will search its Forwarding Information Base (FIB) to longest match this interest's name and find the egress interfaces to forward this Interest to the next routers. If this Interest can not be satisfied by the routers along its transmission path, it will reach the corresponding data provider (data producer). (3) When the data packet is sent back, the router finds the corresponding PIT entry and forwards the data packet to all the ingress interfaces listed. In addition, the router removes the corresponding PIT entry, and caches the data in its CS.

2.2 Flajolet-Martin (FM) Sketch and its Alternatives

In order to estimate the number of distinct items in a large set, many approaches for the heavy-hitter problem have been extensively studied. Among them, counter-based approaches maintain a bounded-size cache, and leverage some efficient algorithms, such as MG algorithm [20], to count the number of distinct items in the set. However, each monitored item needs its own counter, and these counters need to be updated whenever a new item is inserted [21]. Considering the overhead for monitoring items and updating the corresponding counters, they can only be used in a small portion of the counting applications [22]. To achieve a light-weight solution, probabilistic sketch-based approaches [23], [24], [25], [26], [27], [28] are proposed. At a small cost of counting accuracy, this type of approaches outperforms the existing counter-based approaches in terms of counting overhead and efficiency. Among them, FM sketch and its alternatives [23], [27], [27], [28] outperform the other existing schemes in terms of computational and spacial overhead and counting efficiency. It applies a group of functions H_1, $H_2, \ldots, H_k$ to hash all items, and constructs k corresponding bitmaps $A_1, A_2 \ldots, A_k$ to track the hashing results. The length of each bit map is equal to the number of bits of a hash value. The basic procedures to estimate the distinct number of items that arrive sequentially are shown as an example in Fig. 1 and introduced as follows:

1) When receiving the first item S_1, $H_1, H_2, \ldots, H_k$ are applied to hash this item to obtain k hash values.
2) As the feature used to represent each distinct item, the position of the leftmost "1" bit of the hash value of each H_i is inserted into bitmap A_i. In Fig. 1, the position of the leftmost "1" bit of the hash value from H_1 is 2, so the second bit of A_1 is set to "1". Similarly, the left most "1" of the hash value from H_2 is 6 and the sixth bit of A_2 is set to 1. This process continues until the leftmost "1" bits of all hash values for Item S_1 are aggregated into the corresponding bitmaps so the features of S_1 are added into the FM sketch.
3) For the next item S_2, the FM sketch repeats the above operations to insert the leftmost '1" bits of all k hash values for S_2 into $A_1, A_2 \ldots, A_k$. This process continues when a new item arrives, and the bitmaps can record the features of all received items compactly.

By tracking the leftmost "1" bit of the hash values, the bitmaps can capture the probabilistic features of the received items. It is not difficult to see that the leftmost bit of each bitmap has the highest chance of being filled with "1" (50% for the hashing of each item). The second bit from the left has the second chance of being set to "1", and its probability is under the condition that the first bit of the hash value is a "0". The chance of each next bit to be set to "1" bit reduces. In addition, the chance for each bit of a bitmap to be filled with "1" increases as the number of distinct items increases. Thus to estimate the number of distinct items, we can track the length of the sequence of "1" from the left of the bitmap. To determine this length, FM sketch searches for the leftmost "0" bit in each bitmap. To reduce the error rate of estimation, it estimates the result by finding the arithmetic mean of the indices of the leftmost "0" bits of all bitmaps [23].

The number of hash functions and corresponding bitmaps used in an FM sketch have impact on the final estimation accuracy. It generally requires at least hundreds of hash functions and bitmaps to guarantee the accuracy [23]. As the most effective and latest version of FM sketch,

Hyperloglog FM sketch [27] reduces the number of hash operations to one by leveraging the categorized hash-value counting. It forms a group of bitmaps to record the probabilistic features of different groups of hash values. For each hash value, it uses the left several bits as a group index. For the counting purpose, its tracks the leftmost "1" bit of the remaining substring.

Both FM sketch and Hyperloglog FM sketch neglect most probabilistic features captured in a generated hash value, but only consider the leftmost "1" bit of a hash value or the leftmost "1" of the hash bits excluding the part for group index. Although Hyperloglog FM sketch reduces the number of hash operations on each item to one, it needs to collect more items to capture sufficient probabilistic features to estimate a stable counting result, which will result in a long estimation delay.

To timely identify the attack traffic before

2.3 Monte Carlo Hypothesis Test

To learn the criteria for attack identification in an efficient and lightweight way, we study state of the art learning technologies, and choose a re-sampling-based hypothesis test, Monte Carlo hypothesis test, to achieve lightweight yet timely criteria learning. Monte Carlo hypothesis test constructs several repeated tests and achieves an estimation on the underlying distribution according to the test results. To provide sufficient samples for all the repeated tests, each Monte Carlo hypothesis test re-uses the latest set of samples by randomly re-sampling the set without replacement. Compared to a conventional hypothesis test, Monte Carlo hypothesis test needs fewer samples to obtain an estimation on the underlying distribution. A Monte Carlo hypothesis test follows four steps: (1) Performing the initial sampling with S samples taken from the data of interest and obtaining a sampling set; (2) For the re-sampling, randomly selecting a value from the sampling set as a new sample, putting it back to the sampling set so that it has a chance of being drawn again, and continuing this process until totally S new samples are obtained to form a re-sampling set; (3) Computing a statistic estimation for the S members of each re-sampling set [29] and displaying the estimation results from different sampling groups in a histogram to represent the distribution of the estimation results; (4) Setting a level of significance α on the distribution to find a convincing estimation result. Hence, this re-sampling-based hypothesis test technology reduces the number of the required samples as well as the test latency.

3 Threat Model of DDoS Attacks

To detect DDoS attacks in NDN effectively, we first study different types of DDoS attacks in NDN by building threat models for different DDoS attacks in Section 3.1. Based on the analysis on threat models, we discover the adversarial strategy for malicious nodes to performing an effective DDoS attack in NDN, which is verified through extensive simulations of the attack damage effect evaluation in the same subsection. Finally, we derive the attack detection guidelines in NDN, based on a significant attack symptom related to the adversarial strategy.

3.1 Modeling DDoS Attacks in NDN

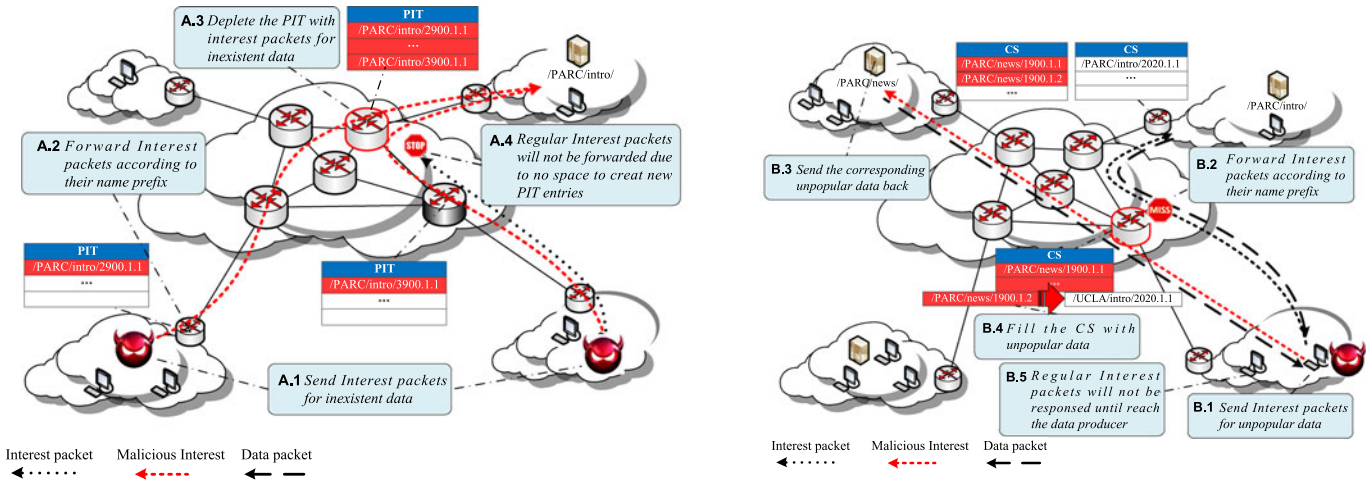
As the primary security consideration of NDN design, NDN can guarantee the integrity of data and thus defend the conventional attacks that aim to modify data packets. However, it is susceptible to another type of primary threats to Internet, i.e., Denial of service (DoS) attacks, that can deplete the resources on NDN routers. The DoS attack, especially, its distributed variants (i.e., DDoS attack), have *plagued* today's Internet by exhausting different types of network resources. In NDN, an adversary cannot send an arbitrary number of interest packets through specific channels to exhaust their bandwidth thanks to the Interest aggregation of the name-based transmission. However, it can take advantage of the vulnerability of PIT or CS to mount new DDoS attacks specific to NDN, among which Interest Flooding Attack (IFA) aims to deplete Pending Interest table and thus block the transmission of the subsequent regular interests, and Cache Pollution Attack (CPA) can violate the data locality of the in-network caches by requesting unpopular data. When a CPA attack is successfully launched, the regular users will not be able to retrieve the expected data from the network caches. Instead, they need to turn to the distant data provider, which will incur higher delay and reduce network throughput.

3.1.1 Interest Flooding Attack

The IFA attack targets to deplete the PIT on a specific router and the subsequent Interests will be discarded since there is no space of the PIT to record the information of these Interests. As depicted in Fig. 2a, to perform an IFA attack, the adversary needs to forge a large number of malicious Interest packets to request different non-existent data (step *A.1*). In step *A.2*, the routers on the path will forward these malicious Interest packets according to the Forward Information Base (FIB) entries that have the longest matched prefixes with the Interest names, and also record the information of these Interest packets in their PITs until the corresponding data packets are sent back. Since the aforementioned Interest packets requested no-existing data, there is no data sent back, so the space of the PITs occupied cannot be released before the time limits of the corresponding PIT entries are reached (step *A.3*). As shown in step *A.4*, if a regular Interest packet arrives at one of these routers, the router can only drop the information associated with the Interest due to the lack of available PIT space. These routers and other regular users using these routers are the attack victims.

3.1.2 Cache Pollution Attack

The malicious users can violate the data locality of the in-network caches by performing a CPA attack [30], [31]. CPA attacks can be characterized into two subtypes, Locality Disruption Attack (LDA) and False Locality Attack (FLA). An LDA attack continuously requests different unpopular data to ruin the data popularity distribution (see Fig. 2b). In step *B.1*, the adversary makes the malicious users to forge a large number of malicious Interest packets, requesting different unpopular data. In step *B.2*, these Interest packets are forwarded by the routers on the transmission path. In step *B.3*, after receiving a malicious Interest packet, the data provider sends the corresponding unpopular data back along the



(a) Interest Flooding Attack

(b) Cache Pollution Attack

Fig. 2. DDoS attacks in NDN.

reverse path of the Interest packet. This unpopular data will be cached in the content store of the routers on the transmission path, and squeeze the cached popular data out of the content store (step *B.4*). As a result, the subsequent regular Interests requesting popular data cannot be satisfied by the cached data in the content store of the routers on the transmission path, and thus will be forward to the corresponding data providers (step *B.5*).

An FLA attack repeatedly requests the same set of unpopular data to create a false data popularity distribution at the CS of each router on the transmission path. However, NDN routers will aggregate the Interest packets from different incoming interfaces, and discard the Interests requesting the same data. Therefore, the FLA attacker can hardly distort the popularity of the cached data on a router unless the unpopular data are requested from different interfaces on this router, i.e., the malicious Interests must be requested by users at different parts of the network and forwarded along different transmission paths. As the attacker can hardly arbitrarily compromise the users at the different parts of the network. it is difficult to launch a FLA attack in NDN.

3.2 Adversarial Strategy Analysis

According to the definitions of IFA and LDA attacks, the compromised network nodes must send malicious Interest packets to perform any of these attacks. In a real-world network, a large number of network nodes will forward packets to a specific router in a short time period. If the number of compromised nodes is small, the number of malicious packets and their impacts are also small. If a compromised node dramatically increases the frequency of forging and forwarding malicious Interest packets, the resource of its downstream next-hop router will be exhausted and fail to support forwarding the subsequent malicious Interest packets. Hence, it requires the existence of a lot of compromised nodes in different subnets to perform an effective denial-of-service attack in NDN.

To perform an effective distributed denial-of-service attack, the compromised nodes should target a router on the common transmission path of their attack packets. In general, these compromised malicious nodes do not know where a data provider indicated by a name prefix is in a NDN network. Therefore, they cannot make the malicious packets forwarded towards the same router according to their addresses. They can only attach these malicious packets with a common name prefix that they are forwarded to the same target according to the longest name prefix match principle. *This ia an inevitable adversarial strategy for malicious users to mount an effective DDoS attack in NDN.* Here, the name prefix that the compromised nodes use to form the names of the malicious packets is named as an attack prefix.

On the contrary, different from the malicious traffic of DDoS attacks, the regular interests normally request a group of data under a Zipf-liked distribution, rather than a significant increase of the number of requests for the distinct data. Therefore, *as an important traffic pattern of DDoS attacks, a victim router can observe a dramatic increase of Interests that request for distinct data using a common name prefix (attack prefix).*

3.3 Adversarial Strategy Verification

To investigate whether the discovered adversarial strategy is the inevitable choice for any adversary to perform an effective DDoS attack in NDN, we evaluate the damage effect of all two types (*e.t.* IFA and LDA) of DDoS attacks by using different number of attack prefixes.

3.3.1 Simulation Setup

The processes of these DDoS attacks are evaluated in ndnSIM [19], the most extensive NDN simulator. All of our simulations were performed on a local machine, equipped with an Intel Core i7 3.4G CPU and 16G RAM, running Ubuntu 15.04 with kernel version 3.19. All the primary parameters used in ndnSIM are listed in Table 1.

The PIT size of each router is 15000, and the cache size is 1000 (which is 1% of the total number of distinct data used in the simulations). Since a CPA attack can significantly affect the in-network caches regardless of the types of

TABLE 1
Simulation Parameters

Simulation Parameters	Value
# Data	100,000
Link (Edge nodes to Gateway)	100Mb/s
Other link	1000Mb/s
CS strategy	LRU
CS size	1000 data items
Maximum PIT size	15000 entries
Size of each data	1,024 bytes
Rate of regular traffic	3000 Interests/s
Zipf power of regular traffic	0.7
Life time for Interests	20s

replacement strategies [6], we chose LRU (least recently used) without loss of generality.

To simulate the background traffic, each benign user sends 3000 regular interests per Second, requesting data by following a Zipf distribution with a power parameter 0.7. The name of an interest consists of a name prefix and a distinct random number. Specifically, five optional name prefixes, "/google", "/amazon", "/youtube", "/yahoo" and "/facebook", are used throughout our simulations.

3.3.2 Effectiveness of Adversarial Strategy

We simulate all types of DDoS attack sceneries over a real-world ISP network (see Fig. 3), Telstra, the network topology of Australian ISP. It includes 65 backbone routers, 35 gateway routers, and 169 edge routers. On this topology, we randomly deploy five compromised users on different edge nodes. To verify the effectiveness of adversarial strategy, we make these compromised users use different number of attack prefixes to forge Interest packets and perform attacks from Second 2. First, all of compromised users take "/amazon" as the attack prefix to perform attacks. Second, compromised users are divided into 2 groups, one with two malicious users using "/amazon" to perform attacks and the other taking "/youtube" as the attack prefix. Finally, each compromised user has its own choice of attack

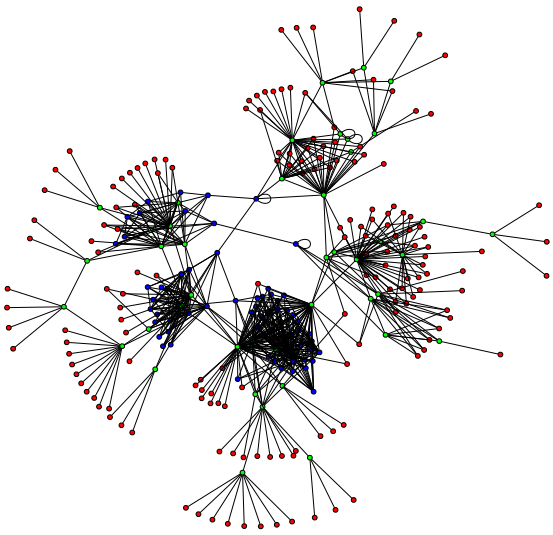
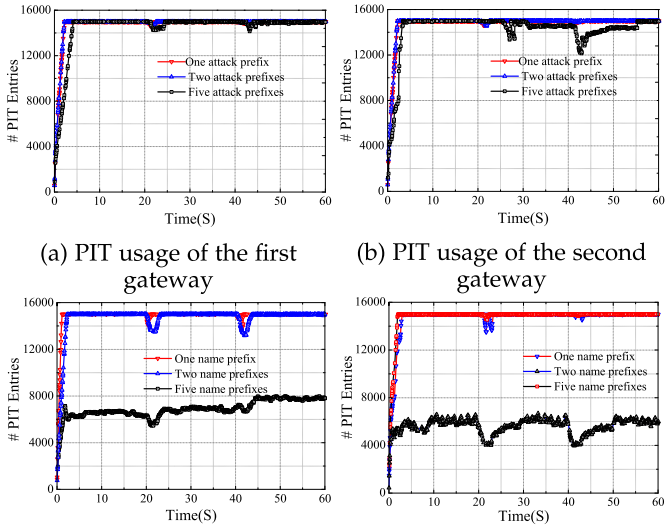


Fig. 3. Simulation topology.



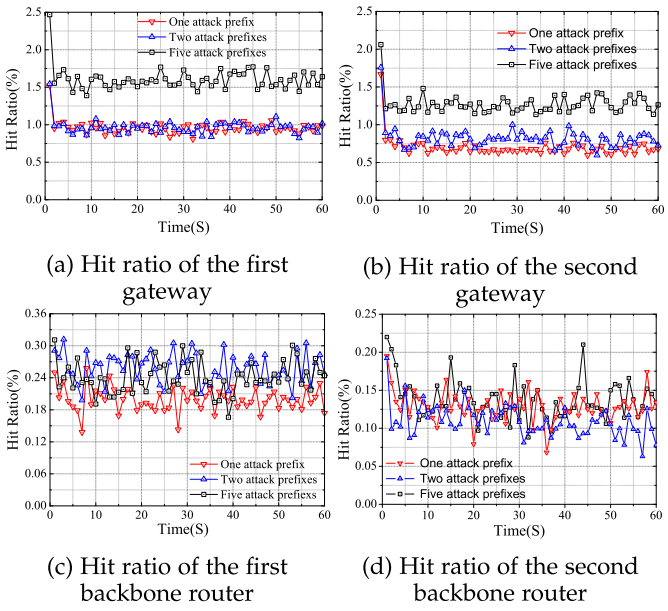
(a) PIT usage of the first gateway
(b) PIT usage of the second gateway
(c) PIT usage of the first backbone router
(d) PIT usage of the second backbone router

Fig. 4. PIT usage under IFA attacks.

prefixes, including all name prefixes used for generating background traffic. Additionally, the compromised users will extend the attack prefix with random numbers in a range different from those used by regular interests. With the similar configuration of the benign users, each compromised user sends 3000 malicious interests per second.

For IFA attacks, the PIT usage of network nodes in different topology places is depicted in Fig. 4. Two types of network nodes on the crucial path (two gateways of providers, and two backbone routers) are involved into the evaluation. We have four observations about PIT usage of these network nodes under IFA attacks: (1) The PIT usage of these network nodes reaches 15000, the maximum value of PIT, when malicious users share the same attack prefix. (2) On the gateways, the malicious traffic of the attacks using one and two attack prefixes raises the PIT usage to its maximum value within 3 seconds, and the attacks using five different prefixes make the PIT usage reach the maximum value in the fifth second. A part of malicious Interest packets can converge in the gateways in all cases. (3) On the backbone routers, the attacks using one and two attack prefixes cause the same trend of the PIT usage. In contrast, the number of the used PIT entries increases to almost 8000 and fluctuates below 8000 when the attack uses five different attack prefixes. It appears that most of malicious Interest packets bypass these backbone routers when the malicious users take their own name prefixes.

In Fig. 5, we obtain similar observations on the damage effect of CPA attacks. The attacks with only one attack prefix reduce the hit ratio of the cache on different network nodes most severely, while the attacks with more attack prefixes create moderate damage effect the routers. The damage effect on the PIT and Cache causes the longer data transmission latency for the data possessing the name prefix of "/amazon", as listed in Table 2. For the attack with one attack prefix, the average evaluation results of data transmission latency and their standard deviations are degraded most seriously. The transmission latency under the attack with five different attack prefixes is almost equal to that



(a) Hit ratio of the first gateway

(b) Hit ratio of the second gateway

(c) Hit ratio of the first backbone router

(d) Hit ratio of the second backbone router

Fig. 5. Hit ratio under CPA attacks.

without the interference of attacks, which also is 0.2 second. We conclude that only the attacks with one attack prefix can create the severe damage effect on the victim routers.

4 Guidelines for Detection Design

Our studies (in Section 3.2) have shown that victim routers suffering from DDoS attacks in NDN network will observe a common traffic pattern, a dramatic increase of the number of Interests that possess the same name prefix (attack prefix) but request distinct data contents. We will exploit this attacking pattern for detecting both Cache Pollution attacks and Interest Flooding attacks in NDN.

4.1 A Straw-Man Scheme and its Limitations

According to the traffic pattern of DDoS attacks, the routers can identify the name prefix used by attackers and block their subsequent malicious requests. A router can detect a DDoS attack if there is a dramatic increase of distinct data requests which possess a common name prefix (Section 3.2). This traffic pattern can be exploited to detect DDoS attacks following two processes, traffic monitoring and attack identification.

In the traffic monitoring process, a router watches for the burst of Interest packets that possess a common name prefix. If the burst exceeds with an empirical threshold, a potential DDoS attack is identified and the corresponding common name prefix used by the malicious Interest packets is reported immediately and the Interest packets possessing this name prefix will be blocked. A router can simply rely on FM sketch or its alternatives to monitor the Interest packets which possess a common name prefix. In the attack identification phase, the router periodically check the interest traffic monitoring result (i.e., the counting result of the sketch), and compares the result with an empirical threshold. If the result exceeds the threshold, the router reports a potential DDoS attack immediately.

Limitations. If FM sketch or its alternatives is used to monitor Interest traffic, the straw-man scheme needs to use a large set of hash values to capture sufficient probabilistic features and achieve a stable traffic monitoring result, which will consume more computation and storage resources and delay the whole attack detection process. Even worse, since a name prefix can be composed of various strings, there are a huge number of candidates to be used as the attack prefix, which would require an extremely large number of FM sketches to monitor Interests with different name prefixes. The computation and memory resource consumption in the traffic monitoring process will increase dramatically.

TABLE 2
Latency Under IFA and CPA Attacks

	Metric (S)	#Attack prefixes		
		One	Two	Five
IFA	Expectation	8.90	5.59	0.19
	Standard Deviation	9.53	7.74	0.45
CPA	Expectation	1.53	0.57	0.20
	Standard Deviation	3.48	1.89	0.57

4.2 Detection Guidelines

The simple straw-man scheme, however, is not efficient enough due to only using a small fraction of the information in each hash value and the large number of redundant hash values generated in the FM sketch and its alternatives. We study the primary issues to address, and provide a group of reasonable design guidelines.

A persistent DDoS attack detection solution should be lightweight to avoid consuming too much additional resource and overload the victim. Among the two aforementioned processes of DDoS attack detection, the traffic monitoring process will consume much more resources of routers in order to check Interest packets continuously, while routers only need to learn and use a threshold for identifying abnormal increment of the distinct data requests with common name prefixes in the initial phase. Except for the need of low resource consumption, the attack detection should be efficient to timely capture the traffic pattern of the attack. We provide the corresponding guidelines for traffic monitoring process and attack identification process respectively, as the following:

Guidelines for Traffic Monitoring Process:

1) There is a need to eliminate redundant resource consumption in FM sketch.
2) The Interest requests should be monitored continuously for timely detecting the attack and blocking the attack as soon as possible.

Guidelines for Attack Identification Process:

1) The threshold learning process should be limited in a short time period.
2) To avoid excessive resource consumption and attack identification latency, we pursuit an unified DDoS attack detection solution by monitor all Interest packets with different prefix names, instead of monitoring Interests for each specific name prefix with a standalone sketch.

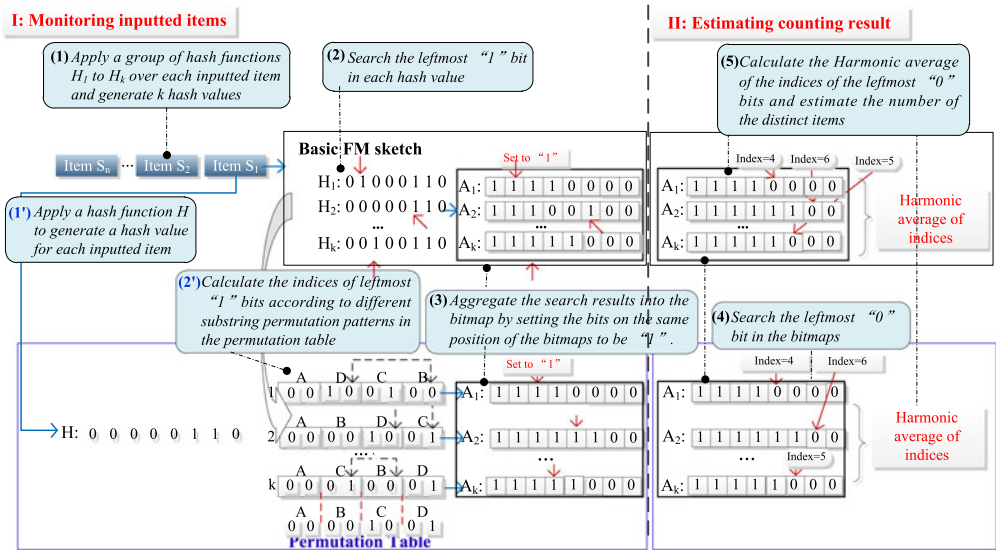


Fig. 6. Reduce resource consumption of the sketch-based traffic monitoring.

5 LiEffi-FM Sketch: A Lightweight and Efficient FM Sketch

To timely identify potential DDoS attacks, routers need to continuously monitor the Interest packets. To reduce the resource consumption and also to speed up the detection, we propose a novel **Li**ghtweight and **Effi**cient **F**lajolet-**M**artin sketch (LiEffi-FM Sketch) for traffic monitoring. Its design follows the guidelines for traffic monitoring in Section 3.1.

5.1 Reducing Resource Consumption

Rather than only using a small fraction of the information in each hash value, we propose a novel counting strategy by fully utilizing all bits of a hash value to reduce the number of hash operations thus the large overhead in obtaining an accurate counting result. More specifically, we permutate a given hash value K times to replace K hashing operations of FM sketch. For each permutation, we treat it similarly as a new hash value, and use its leftmost "1" in the sketch operation.

Fig. 6 illustrates our proposed schemes. For an item to insert into the sketch, we first use a hash function to generate a hash value, and then generate a large number of permutations. In Section 7, we will show that the use of permutation can achieve a counting accuracy similar to that of using multiple hash values. Rather than permutating all bits, we divide a hash value into substrings, and permutate the substrings. This allows us to use a permutation table in advance to guide the "virtual" permutation searching without consuming high computation and storage resources to physically permutate a hash value. We assign each substring an ID, and for each permutation, we could provide a sequence of sub-string IDs in a row of the permutation table to guide the searching of the leftmost "1" bit in sub-strings. Following we explain the idea through the example in the Fig. 6.

1) For an incoming item "1900.1.1", we first hash "1900.1.1" to generate an 8 bit hash value "00000110". We then split this hash value into 4 substrings, "00", "00", "01" and "10" respectively, and name them as A, B, C, D. To facilitate the finding of leftmost "1" index of a virtual permutated string, we further obtain the index of the leftmost "1" bit in each substring to generate an index set {-1, -1, 1, 0}, where -1 means there is no "1" bit in a substring.
2) For a permutation following the first pattern "ACBD" provided by the permutation table, to find its leftmost "1" bit, we don't need to perform the actual permutation (i.e., "00010010"), but just need to go through the leftmost "1" indexes of substrings in the sequence of A, C, B and D. We first check the leftmost "1" index of A. It is -1, there is no "1" bit in A, and thus we skips A. We continue this check, until we find a substring that has "1" bit, and we count the index of this "1" bit in "ACBD" as the obtained leftmost "1" bit. In this example, the next substring C contains "1" bit, and the leftmost "1" index in "ACBD" is 3, equal to the sum of the length of the skipped substrings (2 bits of A) and the index of the "1" bit in C.
3) We set the bit at index 3 (the fourth bit) in the first bitmap to 1 in this example, without the need of searching the remaining substrings of the permutation (B and D).
4) We continue the above steps on the other permutation patterns in the permutation table, and aggregate their left most "1" bits into the corresponding bitmaps. To obtain the counting result, we calculate the average of the indexes of the leftmost "0" bit from k bitmaps to estimate the number of distinct items.

In order to obtain stable estimation result, we need to monitoring enough packets over a period of time. In Section 5.2, we will introduce our sketch to enable no interrupted estimation. A detailed description of the aforementioned counting process is provided in Algorithm 1. On line 2, a new item is hashed, and the generated hash value is split into M substrings. We find the leftmost "1" bit in each substring and store the results into an index set (line 4-5). Line 7 begins a loop to calculate the index of the leftmost "1" bit in each

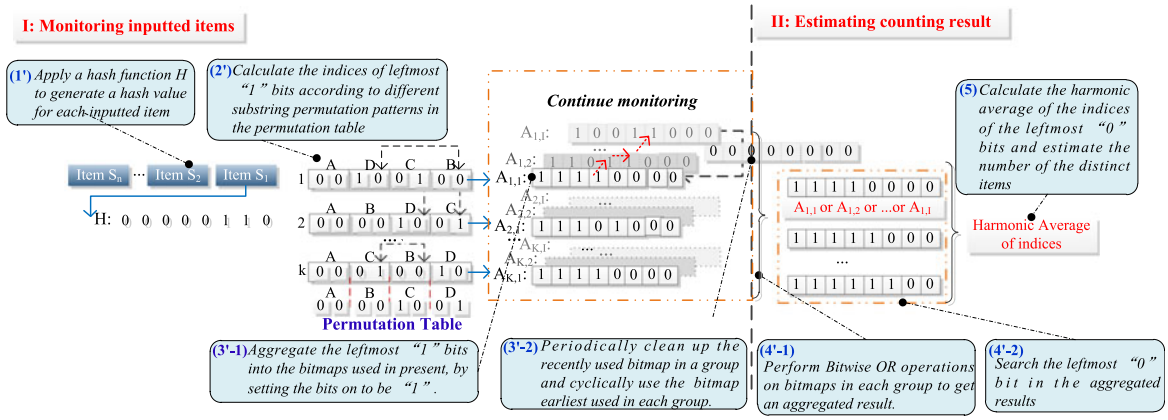


Fig. 7. Continue traffic monitoring based on the sketch.

permutation (line 8-17). We aggregate this result into the corresponding bitmap (line 18) by setting its corresponding position to 1. To obtain the counting result, we search for the leftmost "0" bit in all the bitmaps (line 21) of a time period, and calculate the harmonic mean of the corresponding indexes to estimate counting result (line 22). In our detection, we apply this estimate the number of distinct name suffixes.

Algorithm 1. Basic Counting Algorithm

Input:
T: a new item;
PM[][]: a permutation table, including K rows, where each row contains M elements;
Output:
Est: the estimation of number of the distinct items;

```
1: while The counting period is not end do
2:   Hash over T to get a hash value h of size L;
3:   Split h into a set of M substrings;
4:   Search the index of the leftmost 1 bit in each substring;
5:   Store the search results into an index set p[];
6:   Construct K bitmaps, bitmap[i], 0 <= i <= K − 1;
7:   for i = 0 to N − 1 do
8:     offset = 0;
9:     for j = 0 to M − 1 do
10:      id = PM[i][j];
11:      if p[id] != -1 then
12:        offset = offset + L/M;
13:      else
14:        k = offset + p[id];
15:        Break;
16:      end if
17:    end for
18:    Set the k-th bit in bitmap[i] to be 1;
19:  end for
20: end while
21: Search the index of the leftmost 0 bit in each bitmap;
22: Est is the harmonic mean of the search results;
23: Return Est;
```

To reduce the load of checking each Interest packet to determine if it is a new item, we reuse a hash value in LiEffi-FM Sketch to reduce the number of hash operations to only one. This significantly reduces the computation complexity in the detection of DDoS attack in NDN. This design is based on the observation that the shift operation on a bit string, a bit array or a binary numeral is a simple and fast action, which is directly supported by CPU. In practice, to optimize the performance, we can implement our algorithms with machine-codes according to different CPU instruction sets and operating systems.

5.2 Continuous Estimation With Cyclic Memory Management

In order to timely detect the DDoS attack, the monitoring of Interest packets needs to be performed continuously without interruption. However, as time goes on, the bitmaps will be filled more and more. The increase of diverse items over time is natural but may not reflect the real pattern of the current traffic. The Interests that are checked in the long time back may be outdated and lead to false detection decision, so outdated records should be removed from the bitmaps. However, the current bitmap design does not differentiate between items arriving at different time. If the sketch is simply cleaned with all bitmaps reset to 0, it needs to wait for a period of time to rebuild the stable statistics. If a DDoS attack happens during this initial feature accumulation period, it will create significant damage.

To timely detect DDoS while not being impacted by old records, we propose the use of a cyclic bitmap scheme with time slicing. Rather than aggregating all hashing results into one group of bitmaps, we consider using I time slices, each is associated with K bitmaps as done above. When reaching a new time slice, the bitmaps associated with the oldest time slice will be cleaned up to record the new Interests. The bitmaps used in I slices that are associated with the same row of the permutation table are categorized into one group, and K bitmap groups are formed accordingly. In order to get a stable statistics without being impacted by the short duration of a time slice, the bitmaps in in a group are merged generate a record to support stable statistics (see Fig. 7).

Algorithm 2 depicts how to circularly apply bitmap groups to counting distinct items. On line 1, a new item is hashed, and the corresponding K leftmost "1" indexes are obtained by calling Algorithm 1. We switch the bitmap group used just now (line 3-9), where variable "used" is the index for the group of bitmaps used currently. This variable will point to the first bitmap group if the time slice when the last bitmap group is used ends; otherwise, the next

bitmap group will be used. From line 10 to line 12, the leftmost "1" bits of the new item are aggregated in the K bitmaps in the group indicated by the variable "used". To obtain a counting result, we merge the bitmaps in a group by performing bitwise "OR" operation on them, and search the leftmost "0" bit in the obtained result (line 14-21). In line 22, we calculate the harmonic mean of the leftmost "0" bit indexes to estimate the number of distinct items.

Algorithm 2. Continuous Counting Algorithm

Input:
T: a new item;
Output:
Est: the estimation of the number of distinct items;
1: K bitmap groups, each group includes I bitmaps, totally, $K \times I$ bitmaps, bitmap[i,j], where $0 <= i <= I-1, 0 <= j <= K-1$;
2: Hash over T to obtain a hash value, get K leftmost "1" places, $p[j]$, by applying Algorithm 1, recorded into the K bitmaps used in present;
3: **if** A new time slice is coming **then**
4: **if** used==I-1 **then**
5: used=0;
6: **else**
7: used++;
8: **end if**
9: **end if**
10: **for** j = 0 to K-1 **do**
11: Set the $p[j]$-th bit in $bitmap[j, used]$ to be 1;
12: **end for**
13: **if** A counting result is required **then**
14: **for** j = 0 to K-1 **do**
15: **for** i = 0 to I-1 **do**
16: Bitwise OR on bitmap[j,i] to obtain amap[j];
17: **end for**
18: **end for**
19: **for** j = 0 to K-1 **do**
20: Search the index of the leftmost 0 bit in amap[j];
21: **end for**
22: Calculate the harmonic mean of the search results as Est;
23: **end if**
24: **return**Est;

The cyclic design allows LiEffi-FM sketch to timely detect the DDoS without the need of the initial accumulation of statistics and any heavyweight management of the storage.

6 Detection Scheme Design

To make our attack detection scheme light weight and timely, we propose a comprehensive attack detection scheme based on our LiEffi-FM sketch to efficiently identify the attack prefix and throttle the corresponding attack traffics. LiEffi-FM sketch is first used to monitor the increase of the number of recent requests for distinct data that possess a common name prefix. Based on the traffic monitoring results, we learn a threshold for the monitoring results of regular Interest traffic to request data with a common name prefix, and thus can use the learnt threshold to discover abnormal traffic. Note that the name prefix used by an adversary would include various name fragments connected with slashes to forge malicious Interest packets. There are a large number of candidates for the attack prefix, all of which will be monitored during DDoS attack detection in NDN.

6.1 Efficient Learning of the Criteria for Attack Identification

To identify an DDoS attack in NDN, a threshold should be learnt for identifying the abnormal increment of Interest packets that are sent recently to request different data items and the requests share a common name prefix. We learn the threshold according to the traffic monitoring results related to different name prefixes. Once the number of distinct data requested (with a common name prefix) exceeds the predetermined threshold, the common name prefix used in the request packets is reported as an attack prefix. Since Interest traffic is affected by rough time-of-day pattern, long-term trends, and topology differences [32], we should use an adaptive rather than a fixed threshold. In a running network, we should reduce the time taken to learn the change of threshold.

According to Chebyshev's inequality, the traffic monitoring results will fluctuate around their expectation under an upper bound with a high probability [33]. Thus, we can estimate a proper threshold for regular Interest traffic monitoring results. Specifically, we conduct Monte Carlo hypothesis tests on the regular traffic monitoring results acquired in the latest S time slices, and obtain an upper bound of monitoring results that can be used as a detection threshold. If the monitoring results related to a specific name prefix exceed the threshold, a DDoS attack is detected, and the name prefix is reported as the attack prefix. The subsequent Interest packets possessing this prefix should be throttled to defend against the detected attack. This threshold estimation process has three steps:

1) The S traffic monitoring results are taken as the original sampling set, $\{x_i\}$. These monitoring results are randomly drawn from the original sampling set without replacement, in order to build a re-sampling set of size S. The expectation and variance of sample values are calculated for this set.
2) On all other re-sampling sets, we repeat this operation. The distribution of the expectation values as well as that of the variance values is gained. In this way, with a specific level of significance α, we achieve the stable estimations of the expectation and the variance.
3) Finally, we find the upper bound of the sample values according to Chebyshev's inequality, $P(|x_i - E(x_i)| \geq \epsilon) < Var(x_i)/\epsilon^2$, where x_i is an arbitrary sample in the re-sampling set, $E(x_i)$ and $Var(x_i)$ are the estimations of expectation and variance of sample values, respectively. ϵ is the upper bound, and used as the threshold.

Since the Monte Carlo hypothesis test can be performed on a small group of samples, our learning process of the detection threshold can be performed after collecting very few traffic monitoring results. Additionally, to adapt the threshold to current Interest traffic, we perform the above threshold estimation process after the empirical monitoring results deviate from the threshold multiple times almost

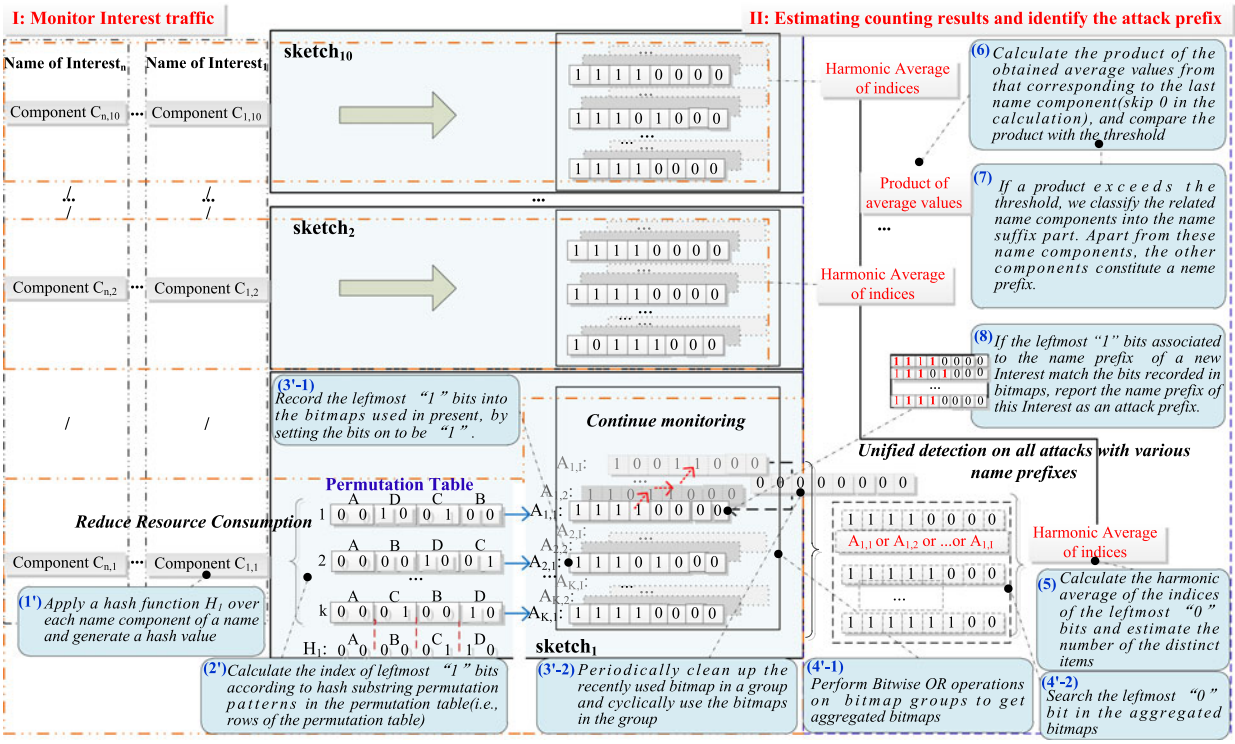


Fig. 8. Monitoring interest name with multiple LiEffi-FM sketches.

uniformly over a long period of time (e.g., 6 hours), besides when a router is initiated. In detail, how many times and how long the empirical monitoring results deviate from the threshold before learning a new threshold should be configured according to the practical configuration experience. A fixed updating schedule of the threshold is vulnerable for the well designed attacks.

6.2 Towards Persistent Attack Detection

As a prefix may consist of various strings (name components) separated by slashes, to identify the attack, a straightforward way is to assign a sketch to each prefix and look for the abnormal increase of the related Interest packets that request for distinct data [18]. However, there are numerous name prefixes (with different combinations of name components), it is impossible to have a sketch to monitor each group of interests possessing a specific name prefix. Besides the storage overhead, the overhead to match every name prefix is prohibitive.

If an LiEffi-FM sketch is used to monitor Interest packets, the probabilistic features of the packets will be recorded in the bitmap groups. Compared with regular Interest packets, malicious ones will share a common name prefix but possess vary different name suffixes. In the case that different sketches are used to monitor the name prefix and suffix of Interest packets respectively. If the Interests are malicious, the bitmap groups of the sketch for monitoring the name prefixes of these Interests will share the similar "1" bit distribution in bitmaps, while the bitmap groups associated with the other sketch for recording the name suffixes get totally different "1" bit distribution in their bitmaps. The "1" bit distribution of the first sketch can be used to identify the attack prefix. More specifically, the name prefix of a new received Interest will be compared with the "1" bit distribution in the bitmaps of the first sketch to determine if the prefix is the attack prefix. A more substantial challenge arises here. A name prefix always consist of multiple name components separated by slashes, and it is a challenge to determine which parts constitutes the attack prefix. Therefore, we need to address three critical issues, separating the prefix and suffix in the names, reporting an attack and identifying the specific attack prefix. We list our procedures as follows.

6.2.1 Sketch Allocation

In Fig. 8, rather than monitoring different name prefixes, we separate a prefix into different components based on "/" and use a fixed number of LiEffi-FM sketches (10 sketches) to monitor the first ten of them. To prevent the failure when the number of name components is larger than the number of sketches, we use the last sketch to monitor the 10th component and any additional name components as a whole string.

6.2.2 Separation of Interest Names and Attack Reporting

Malicious packets normally have very different name suffixes but an unchangeable name prefix. Within the attack process, only the name components included in the name suffix part have a large number of variable values. Therefore, the component with a large number of variable values is statistically more likely to form the name suffix, instead of the attack prefix. To determine which name components constitute the attack prefix, we will neglect these name components. Sketches are used to count diverse candidates for the last several components. If an unusual increase of the

diverse candidates for these components is observed, these components and its successors are grouped into the suffix part (Step 6, 7 in Fig. 8). This process continues until adding more components will not lead to an unusual increase of diverse candidates of the name suffixes. All name components in the suffix part have been found. The remaining components constitute the attack prefix (Step 8 in Fig. 8). More specifically, we take the product of the estimation results of the sketches associated to the last several components as a unified metric, which is used to compare with the detection threshold. Once finding the product exceeds the threshold, an attack is reported, and the corresponding components are grouped in the suffix part.

6.2.3 Identification of the Specific Attack Prefix

While monitoring different packets, the sketch has stored the probabilistic features of their names (”1” bits) in the bitmaps. We can use these ”1” bits as an indicator of the monitored names. Assuming there are g name components grouped into the prefix parts. We look up the sketch associated with each of these g name components to find the common ”1” bit distribution of different bitmap groups. This common distribution is a stable indicator. We cascade the indicators associated with these g name components to form an indicator to identify the specific attack prefix. In detail, we first separate the name of an incoming Interest packet into a prefix and a suffix. While inserting all name components of this Interest packet into the corresponding sketches, we cache the leftmost ”1” bits obtained on the hash results of the prefix components, and then use these ”1” bits to match the ”1” bit distribution of the indicator. If the distribution is matched, the name prefix of this Interest packet has a high probability of being the attack prefix (Step 8 in Fig. 8).

In summary, our proposed detection scheme benefits from the traffic records in LiEffi-FM sketch to timely identify the malicious name prefix. Designed to be persistent, it can be used to continuously prevent against various types of DDoS attacks with very low resource consumption.

7 Analysis

7.1 Security Analysis

The security of the proposed DDoS attack solution is captured by the following lemmas, theorems and analysis.

Theorem 1. *The replacement of original hash values by the permutations of a hash value has no affect on estimating the accuracy of the sketch.*

Proof. Existing FM sketch algorithms rely on a hypothesis that each hash value is totally random, uniformly distributed over the set of binary strings, i.e., every bit in this hash value will be 0 or 1 with the same probability [23]. When using a hash function to generate a hash value (i.e., a binary string), an original input is first encoded in a binary string, and partitioned into a number of substrings. Certain linear transformations are then performed on these substrings to propagate randomness in order to generate a more random hash value. The capability of a linear transformation to propagate randomness is measured by the number of its branches, which are non-zero substrings that remain after the transformation is performed [34].

The number of branches related to a transformation F is calculated by $min_{h\neq 0}(W(h)+W(F(h)))$, where h is the binary string corresponding to an input, $W(h)$ counts the number of its non-zero substrings. Since a permutation operation on substrings only changes the order of its substrings. $W(F(h))=W(h)$, and its number of branches is $min_{h\neq 0}(2\cdot W(h))$, equal to the number of branches for the original hash operation. Therefore, the permutation operations will not reduce the randomness of a hash value, and our replacement of original hash values with the permutations of a hash value has no affect on the accuracy of the sketch estimation. □

Theorem 2. *For LiEffi-FM sketch, the leftmost ”1” bit stored in the bitmaps can indicate the existence of items with a high probability.*

Proof. The probability that an item can be indicated by one unique bit in a bitmap of LiEffi-FM Sketch is equal to:

$$1-\sum_{i=1}^{M}\left(\left(\frac{1}{2}\right)^{\frac{L}{M}\cdot(i-1)}\right)^2\cdot\sum_{j=0}^{L/M-1}\left(\left(\frac{1}{2}\right)^j\cdot\frac{1}{2}\right)^2$$
$$=1-\sum_{i=1}^{M}\left(\left(\frac{1}{2}\right)^{\frac{L}{M}\cdot(i-1)}\right)^2\cdot\left(\sum_{j=0}^{L/M}\left(\left(\frac{1}{2}\right)^j\cdot\frac{1}{2}\right)^2-\left(\frac{1}{2}\right)^{(2L/M+2)}\right)$$

The above probability can be reduced to:

$$1-\frac{1}{3}\left(1-\left(\frac{1}{2}\right)^{2L}\right)+\left(\frac{1}{2}\right)^{(2L/M+2)}\frac{1-(\frac{1}{2})^{2L}}{1-(\frac{1}{2})^{2L/M}}>\frac{2}{3}$$

where L is the length of the bitmap, and M is the number of substrings used to form a permutation.

Since there are K bitmaps used in present, the probability that all these bitmaps can not provide any unique bit to represent an item is equal to $(1-\frac{2}{3})^K$. When K is larger than 512, the probability approaches 0. Hence, for LiEffi-FM sketch, the leftmost “1” bit stored in the bitmaps can indicate the existence of items with a high probability. □

Our Solution can Accurately Detect DDoS Attacks. According to Theorem 2, LiEffi-FM sketch can achieve nearly the same estimation accuracy as FM sketch for the number of distinct items in a huge set. While our solution relies on LiEffi-FM sketch to monitor Interest traffic, different Interest packets will leave an unique record with high probability. In addition, our solution can identify a potential DDoS attack by using a proper threshold generated by Monte Carlo test, whose effectiveness is related to the selected level of significance of Monte Carlo test. Thus our solution is able to detect the increment of distinct data names accurately.

7.2 Performance Analysis

We analyze the performance of the proposed DDoS attack solution, specifically, by evaluating the computational complexity and spatial complexity of LiEffi-FM sketch as follows.

Theorem 3. *The computational complexity for inputting an item into LiEffi-FM sketch is $O(2K)$, where K is the number of bitmap groups used in LiEffi-FM sketch.*

Proof. We analyze the computational complexity of the leftmost "1" bit searching and the bitmap updating as follows. The computational complexity of finding the leftmost "1" bit in a virtual generated permutation consists of two parts. The first part is the computational complexity of finding the leftmost "1" bit in a substring of the hash value. The expectation that the leftmost "1" bit is the h-th bit in the substring is equal to $\sum_{h=1}^{L/M} h \cdot (\frac{1}{2})^h$, where L is the length of the hash value, and M is the number of the substrings. We define a function for formulating this expression:

$$H(x) = \sum_{h=1}^{L/M} h \cdot x^{h-1}.$$

Then we can compute it as follows:

$$H(x) = \left(\int H(x)dx \right)' = \frac{x - x^{\frac{L}{M}}}{(1-x)^2}.$$

Hence, the computational complexity of the leftmost "1" bit search in FM sketch is $M \cdot (2 - (\frac{1}{2})^{\frac{L}{M}-2})$.

The second part is the computational complexity that we compute the index of the leftmost "1" bits in K virtual permutations of the substrings and aggregating them into the corresponding bitmaps. For each virtual permutation, we find the first nonzero hash substring and return its leftmost "1" bit as a result. The computational complexity of finding such a substring is equal to $\sum_{h=1}^{M} h((\frac{1}{2})^{\frac{L}{M}})^{h-1}(1-(\frac{1}{2})^{\frac{L}{M}})$. Therefore, the total computational complexity is equal to:

$$K + K\left(\frac{1}{2}^{L/M} - \frac{1}{2}^{L}\right)\left(1 - \frac{1}{2}^{L/M}\right)^{-1} + M\left(2 - \frac{1}{2}^{L/M-2}\right).$$

Since $(\frac{1}{2}^{L/M} - \frac{1}{2}^{L})(1 - \frac{1}{2}^{L/M})^{-1} < 1$ and $K >> M$, the computational complexity for inputting an item into LiEffi-FM sketch is O(2K). □

Theorem 4. *The computational complexity for estimating the counting result of LiEffi-FM sketch is $O(L \cdot K \cdot I)$, where L is the length of hash value, and I is the number of bitmaps used in I time slices.*

Proof. To estimate the counting result of LiEffi-FM sketch, we merge the bitmaps in each of K groups by performing bitwise "OR" operation on them, and search the leftmost "0" bit in the result. Totally, we perform $K \cdot L \cdot I$ OR operations. In addition, we expect to check $K \cdot (2 - (\frac{1}{2})^{L-2})$ bits to find the leftmost "0" bit in the merging results. Hence, the computational complexity of estimating the counting result of LiEffi-FM sketch is $O(K \cdot L \cdot I + K \cdot (2 - (\frac{1}{2})^{L-2}))$, which is reduced to $O(L \cdot K \cdot I)$. □

Theorem 5. *The spatial complexity of LiEffi-FM sketch is $O(L \cdot K \cdot I)$.*

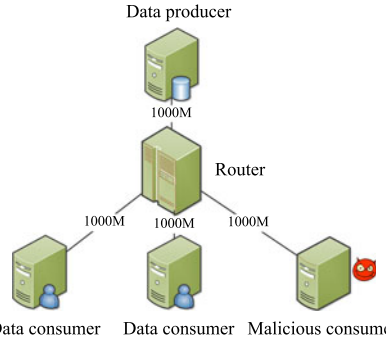


Fig. 9. Experimental network topology.

Proof. The storage resource used in LiEffi-FM sketch includes bitmap groups used in different time slices, a permutation table, and a segment of temporary storage space for caching a generated hash value. Hence, the spatial complexity of LiEffi-FM sketch is $O(L \cdot K \cdot I + M \cdot K + L)$, which can be reduced to $O(L \cdot K \cdot I)$. □

Let δ be the computational complexity of a hash operation. The computational complexity for FM sketch to input an item is $O(\delta \cdot K + K) = O(\delta \cdot K)$, while that for estimating the counting result is equal to $O(L \cdot K)$. The spatial complexity of FM sketch is $O(2 \cdot L \cdot K)$. If we extend it based on the similar technology of cyclic memory management, the computational complexity for the extended version to input an item is $O(\delta \cdot K)$, that for estimating the counting result is $O(L \cdot K \cdot I)$, and the spatial complexity of the extended version is equal to $O(L \cdot K + L \cdot K \cdot I) = O(L \cdot K \cdot I)$. Even compared with the variants of FM sketch, the overall computational complexity and the spatial complexity of LiEffi-FM sketch operations is a smaller constant. *Our sketch-based solution can detect DDoS attacks with limited resource consumption.*

8 Experiments

In this section, we implement our detection solution and a baseline on an experimental NDN network. On the experimental network, we first reproduce the aforementioned DDoS attacks. In addition, we apply the proposed solution to detect these attacks, and compare its effectiveness and resource consumption with those of the baseline.

8.1 Experiment Implementation

We name our proposed attack detection solution as "Persistent" and evaluate its performance on a NFD-based experimental network. NFD [35] is a software defined forwarder that implements and evolves together with the Named Data Networking protocol. The topology of the experimental network is released in Fig. 9, which is a concise version of a real-world network, including a NDN router, a data producer, two data consumers and a malicious DDoS attack node, connected by Gigabit links. Among them, the router is the central node of the network, equipped with an Intel Xeon 2.5G CPU and 16G RAM, which monopolizes two cores of the CPU. The other nodes are similarly equipped, but with 8G RAM and one monopolized CPU core. We install the NFD on Ubuntu 16.04 on all

network nodes. This deployment coincides with the most common attack senior we analyzed in Section 3.

To generate background traffic, each data user sends 1000 regular Interest packets per second. The name of these regular Interest consists of a prefix, "/youtube", and a suffix (i.e., the remaining components of the name except for the name prefix). Different data are requested by Interest packets on a Zipf distribution with a parameter α (0.8), while the total number of different data items is 100,000. Correspondingly, a random integer in the range from 1 to 100,000 is used as the name suffix of regular Interests. The life time of an Interest is set to 20 seconds, after which the corresponding PIT entry will expire and be removed. We configure all nodes according to the transmission scenario of the background traffic. The CS size of each node is equal to 1000, 1% of the total number of regular data packets. Additionally, we limit the PIT size of each NDF node to 3000. Since the CPA attack can significantly affect the in-network caches no matter which replacement strategy is applied [6], we chose LRU (least recently used), which is commonly used by the in-network caches [36]. To evaluate DDoS attacks, the compromised consumer sent malicious interests following the attack strategy proposed in Section 3.1. The name of a malicious interest also consists of a name prefix "/youtube", and a name suffix of a 32 bits random integer larger than 100,000. To launch different types of DDoS attacks, the attack node randomly sends 250, 500 and 1000 malicious Interest packets per second.

Implementation of Our Solution. When implementing our solution, we used murmur hash function [37] to generate hash values due to its efficiency. Each interest name is hashed to a 32-bit hash string, which is further split into 8 substrings. In this way, we are able to estimate the number of distinct data items requested accurately, since we are allowed to include at most 8! permutation patterns in our sketch. In our implementation, we randomly selected 256 permutation patterns to map the features of name components into bitmaps. When using Monte Carlo hypothesis test to obtain the detection threshold, our significance level is 0.5%, and the sampling times is 10. A threshold for the regular traffic is learnt by the router in the first two seconds and fine-tuned to the value based on the weakest CPA attack, so a moderate attack can still be effectively identified.

Implementation of the Baseline. To show the effectiveness of our solution, we also implemented the most lightweight detection approach [10] in the literature as a baseline for comparison. This approach monitors the interest traffic, and calculates the number of each newly requested data. It identifies a CPA attack by checking whether the requesting frequency deviates from a experimental threshold, which was obtained based on the expectation and the variance of the past requesting frequency for these data. In addition, we extend this approach for IFA attack detection, by leveraging the expectation and variance of the number of requested data unsatisfied in 2 seconds to manually select a threshold and detect IFA attacks. Its threshold was also fine-tuned under the CPA attack that sends 250 Interest packets per second.

8.2 Results

We assume the malicious node has been compromised by the attacker and only send malicious Interest packets (i.e.,.

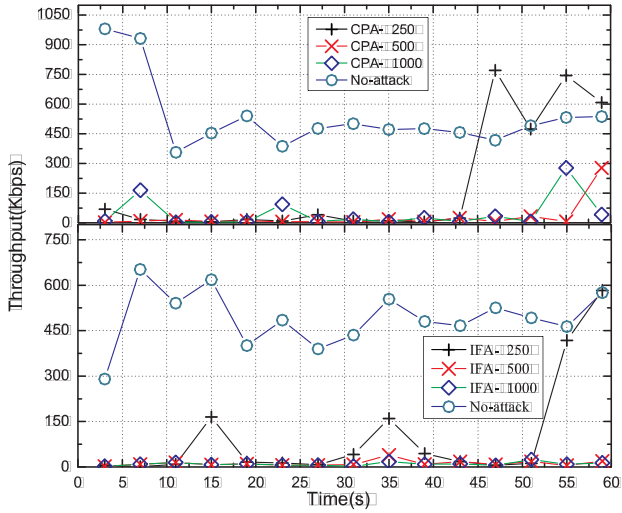


Fig. 10. Damage effect on network throughput.

Interest packets requesting distinct unpopular data and Interest packets requesting nonexistent data). Since the malicious Interest packets and regular Interest packets use different groups of name suffixes, we can distinguish these Interest packets and evaluate the performance of our attack detection solution. From the 2_{nd} second, LDA attacks or IFA attacks are performed by sending incremental number of malicious Interest packets, i.e., 250 Interest packets per second, 500 Interest packets per second and 1000 Interest packets per second. Their damage effects are illustrated in Fig. 10. After these attacks happens, the network throughput drops almost to 0. The throughput remains at a low level in the case of heavy attacks, even when the malicious node stops sending malicious Interest packets at the 50_{th} second. Especially, the IFA attacks deplete the PIT with the Interest packets for non-existent data, and could maintain their damage effects until the corresponding PIT entries expire.

Accuracy. The effectiveness of our attack detection solution is evaluated in terms of detection delay, detection rate, and false positive rate, which are depicted in Table 3. Compared to the baseline, our solution achieves lower false positive rates [10] (FP-R) and similarly outstanding detection rates [10] (D-R). When the malicious Interests have the same rate with the regular Interests (1000 Interests per second), there is no explicit difference between the regular Interest traffic and the malicious Interest traffic. This is the cause of the increase of the corresponding FP-R values. In this case, a more flexible mechanism is desired to block the malicious Interest traffic. Furthermore, the detection delays of our solution (Delay) are always less than 10 seconds. In contrast, the detection delays of the baseline all exceed ten seconds. Both of the solutions obtain relatively high false positive rates when the thresholds were trained to be sensitive enough to capture almost all potential attacks. However, for the attacks that send more malicious Interest packets, this sensitive threshold is fine tuned under the weakest attack and thus may mistakenly consider more regular Interest packets as the malicious ones.

Resource Consumption. To evaluate the resource consumption of our solution, we trace the CPU and memory usage of

TABLE 3
Detection Accuracy Comparison

	Attack	Detection	Delay(S)	D-R	FP-R
IFA	250	Persistent	8.034	95.56%	4.00%
		Baseline	10.970	94.44%	23.00%
	500	Persistent	9.220	95.56%	18.00%
		Baseline	12.539	93.33%	33.00%
	1000	Persistent	9.493	95.56%	28.00%
		Baseline	11.755	94.44%	32.00%
CPA	250	Persistent	8.840	95.56%	2.00%
		Baseline	10.888	94.44%	23.00%
	500	Persistent	9.519	95.56%	21.00%
		Baseline	11.544	93.33%	29.00%
	1000	Persistent	9.392	95.56%	29.00%
		Baseline	11.300	94.44%	33.00%

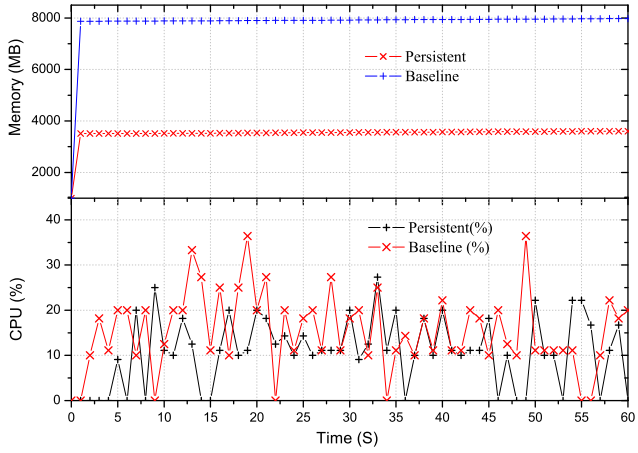


Fig. 11. Resource consumption comparison.

the router under the DDoS attacks that send 1000 malicious Interest packets per second (see Fig. 11). When monitoring Interest traffic, the router deployed the baseline takes more CPU cycles, and requires twice as much memory space as the router with our solution used. The difference on memory usage is due to the difference in the counting of distinct data requests between these detection solutions. The proposed solution estimates the number of distinct requests with a compact way (LiEffi-FM sketch), but the baseline records the recent request frequency of distinct data with a hash table. The difference will be widened under the real network traffic, which possesses a higher variety of name structures, and will take away more memory and computation resource for recording the requested data in terms of their different name prefixes.

9 Related Work

For more complete reviews, we discuss literature studies related work in this section.

9.1 Conventional DDoS Attack Detection

DDoS attack is a major threat in the current Internet [30]. Numerous research efforts [31], [38] have been devoted to mitigate this type of attack. In particular, they analyze the characteristics of the network flows in terms of different IP addresses in a certain period, and thus are able to identify and throttle a malicious client. However, in NDN, we need to analyze the network traffic in terms of hierarchically structured packet names, rather than well-formatted IP addresses. Due to the differences between packet names and IP addresses in nature, we need to re-design the CPA detection mechanism for NDN networks [6].

9.2 DDoS Attacks in NDN and Their Countermeasures

Under an Interest Flooding attack (IFA) [4], a malicious user could attempts to deplete PIT entries with malicious requests for no-existent content [4]. Additionally, the malicious user can violate the content locality of the in-network caches by performing a cache pollution attack (CPA) [30], [31]. In this way, it can cause a large number of cache misses and significantly degrade quality of services for regular users. As the primary DDoS attack threats in NDN, both of these two attacks aim at disrupting NDN Data transmission rather than depleting the transmission channel bandwidth. Actually, the similar types of attacks have been perpetrated in the DNS system [39]. Compared to these attacks, the overhead to detect DDoS attacks in NDN is much higher, considering the extremely large number of data name identifiers used. The special challenge to detect DDoS attacks in NDN is to alleviate this burden without degrading the detection accuracy.

Interest Flooding Attack and its Countermeasures. The primary countermeasures against Interest Flooding attack (IFA) try to limit the number of incoming Interests, either per prefix, or per interface. The main challenge is to distinguish valid from malicious Interests. In [7], Token Bucket is applied for monitoring the ratio of Interest packets and data packets to identify malicious requests for non-existing content in terms of distinct interfaces, whereas Poseidon [8] additionally takes the number of current PIT entries into account for more accurate malicious request detection. These interface-based countermeasures apply thresholds to mitigate malicious request forwarding from an interface but all requests from the throttled interface will be affected by the limitation. However, satisfaction ratio cannot identify potential attacks when considering network congestion[40]. Zhi et al. [41] proposed to use the well-known Gini coefficient to measure a side effect of IFA: the discrepancy in the range of requested content. Similarly, Theil index-based detection is proposed in [42]. The hierarchical structure of the prefixes used by malicious users are exploited in [43]. In [44], to distinguish IFA malicious Interest packets by incorporating data provider, a hash-based signature is attached behind the corresponding Interest name. According to hash-based security labels (HSLs) of name prefixes, InterestFence[45] detects IFAs on the content servers rather than routers. Tan et al. [12], with adapting different attack intensity, develop a generalized likelihood ratio test to detect evolved IFA attacks.

Cache Pollution Attack and its Countermeasures. Although Cache Pollution attack (CPA) will degrade the performance of NDN networks significantly [6], [46], till now, there is very limited work on mitigating NDN CPA attacks in practical. Xie et al. [6] propose a proactive CPA countermeasure

to enhance cache robustness to CPA attacks without any attack detection process. In their work, when a router receives a content packet, it evaluates an exponential function of the requesting frequency to provide the probability that this content packet will be kept in the cache. One disadvantage of their work is, it will prevent caching some potential popular content requested by regular users. Conti et al. [10] propose a lightweight approach for detecting LDA attacks. They first point out that the existing proactive countermeasure [6] is ineffective when defending against realistic adversaries. They then propose their approach based on the hit rate records of the received content. Their solution will consume a large number of computation and memory resources for storing and iterating these hit rate records. Recently, a detection and defense scheme with the help of grey forecast is proposed in [47]. Some other learning-based approaches [11], [14], [15], [16], [17] are also not efficient enough in a realistic network because they all build on top of heavyweight primitives like artificial neural network.

10 Conclusion

NDN is vulnerable to new DDoS attacks, Cache Pollution attacks and Interest Flooding attacks, which degrade NDN transmission significantly. They compromise the crucial components of NDN routers, content stores and pending interest tables, respectively. With a comprehensive study of all types of DDoS in NDN, we discover a common strategy for an adversary to perform an effective attacks in NDN. That is, an adversary needs to send a large number of malicious Interest packets to share a common name prefix, so that the malicious packets will converge at the same network node and disable this victim node. Since these attacks themselves have already imposed a huge burden on victims, to avoid exhausting the remaining resources on the victims, we propose to exploit a lightweight and efficient FM sketch (LiEffi-FM sketch) that can monitor Interest packets persistently. More importantly, LiEffi-FM sketch can be applied to detect various types of DDoS attacks at the same time. The experiments on a real NDN network demonstrate that the introduced detection solution achieves effective and efficient DDoS attack detection only at a little cost of computation and storage resources, therefore, is feasible to be deployed in any practical network and run persistently to eliminate the threats of DDoS attacks in NDN.

References

[1] V. Colella, "Advancing from artificial intelligence to artificial enlightenment," [Online]. Available: https://www.citi.com/ventures/perspectives/opinion/artificial-enlightenment.html

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th Int. Conf. Emerg. Netw. Experiments Technol.*, pp. 1–12.

[3] Z. Lixia et al., "Named data networking (NDN) project," PARC, Tech. Rep., 2010.

[4] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS and DDoS in named data networking," in *Proc. IEEE 22nd Int. Conf. Comput. Commun. Netw.*, 2013, pp. 1–7.

[5] E. Mannes and C. Maziero, "Naming content on the network layer: A security analysis of the information-centric network model," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–28, 2019.

[6] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2012, pp. 2426–2434.

[7] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in named data networking," in *Proc. IEEE IFIP Netw. Conf.*, 2013, pp. 1–9.

[8] A. Compagno, M. Conti, P. Gasti, and G. Tsudik, "Poseidon: Mitigating interest flooding DDoS attacks in named data networking," in *Proc. IEEE 38th Annu. Conf. Local Comput. Netw.*, 2013, pp. 630–638.

[9] H. Dai, Y. Wang, J. Fan, and B. Liu, "Mitigate DDoS attacks in NDN by interest traceback," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2013, pp. 381–386.

[10] M. Conti et al., "A lightweight mechanism for detection of cache pollution attacks in named data networking," *Comput. Netw.*, vol. 57, no. 16, pp. 3178–3191, 2013.

[11] A. Karami and M. Guerrero-Zapata, "An ANFIS-based cache replacement method for mitigating cache pollution attacks in named data networking," *Comput. Netw.*, vol. 80, pp. 51–65, 2015.

[12] T. Nguyen et al., "Reliable detection of interest flooding attack in real deployment of named data networking," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 9, pp. 2470–2485, Sep. 2019.

[13] H. Guo, X. Wang, K. Chang and Y. Tian, "Exploiting path diversity for thwarting pollution attacks in named data networking," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 9, pp. 2077–2090, Sep. 2016.

[14] H. Park, I. Widjaja, and H. Lee, "Detection of cache pollution attacks using randomness checks," in *Proc. IEEE Int. Conf. Commun.*, 2012, pp. 1096–1100.

[15] L. Yao, B. Jiang, J. Deng, and M. S. Obaidat, "LSTM-based detection for timing attacks in named data network," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.

[16] Y. Zeng, G. Wu, R. Wang, M. S. Obaidat, and K.-F. Hsiao, "False-locality attack detection using CNN in named data networking," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.

[17] A. Gupta and P. Nahar, "Detection of cache pollution attacks in a secure information-centric network," in *Data Analytics and Management*. Berlin, Germany: Springer, 2021, pp. 377–397.

[18] Z. Xu, B. Chen, N. Wang, Y. Zhang, and Z. Li, "ELDA: Towards efficient and lightweight detection of cache pollution attacks in NDN," in *Proc. IEEE 40th Conf. Local Comput. Netw.*, 2015, pp. 82–90.

[19] A. Afanasyev et al., "ndnSIM: NDN simulator for NS-3," UCLA, 2012.

[20] K. Mirylenka, G. Cormode, T. Palpanas, and D. Srivastava, "Conditional heavy hitters: Detecting interesting correlations in data streams," *VLDB J.*, vol. 24, no. 3, pp. 395–414, 2015.

[21] G. Cormode and M. Hadjieleftheriou, "Methods for finding frequent items in data streams," *VLDB J.*, vol. 19, no. 1, pp. 3–20, 2010.

[22] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," in *Proc. 28th Int. Conf. Very Large Databases*, 2002, pp. 346–357.

[23] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *J. Comput. Syst. Sci.*, vol. 31, no. 2, pp. 182–209, 1985.

[24] N. Alon et al., "The space complexity of approximating the frequency moments," *J. Comput. Syst. Sci.*, vol. 58, no. 1, pp. 137–147, 1999.

[25] Z. Bar-Yossef et al., "Counting distinct elements in a data stream," in *Proc. Int. Workshop Randomization Approximation Techn. Comput. Sci.*, 2002, pp. 1–10.

[26] M. Durand and P. Flajolet, "Loglog counting of large cardinalities," in *Proc. Eur. Symp. Algorithms*, pp. 605–617.

[27] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm," in *Analysis of Algorithms*. Berlin, Germany: Springer, 2007, pp. 137–156.

[28] S. Heule et al., "Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm," in *Proc. Int. Conf. Extending Database Technol.*, 2013, pp. 683–692.

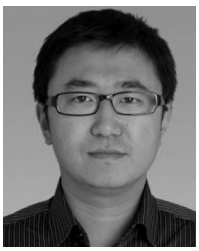
[29] N. Metropolis and S. Ulam, "The Monte Carlo method," *J. Amer. Statist. Assoc.*, vol. 44, no. 247, pp. 335–341, 1949.

[30] R. Gupta and C.-H. Chi, "Improving instruction cache behavior by reducing cache pollution," in *Proc. ACM/IEEE Conf. Supercomput.*, 1990, pp. 82–91.

[31] Y. Gao, L. Deng, A. Kuzmanovic, and Y. Chen, "Internet cache pollution attacks and countermeasures," in *Proc. IEEE Int. Conf. Netw. Protoc.*, 2006, pp. 54–64.

[32] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot, "Long-term forecasting of internet backbone traffic: Observations and initial models," in *Proc. IEEE 22nd Annu. Joint Conf. Comput. Commun. Societies*, 2003, pp. 1178–1188.

[33] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*. New York, NY, USA: McGraw-Hill Education, 2002.
[34] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard*. Berlin, Germany: Springer, 2013.
[35] A. Afanasyev et al., "NFD - NDN forwarding daemon," UCLA, 2018.
[36] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2010, pp. 1–9.
[37] A. Appleby, "Murmurhash 2.0," 2015. [Online]. Available: http://sites.google.com/site/murmurhash
[38] X. Zhuang and H.-H. S. Lee, "A hardware-based cache pollution filtering mechanism for aggressive prefetches," in *Proc. IEEE Int. Conf. Parallel Process.*, 2003, pp. 286–293.
[39] S. L. Feibish, Y. Afek, A. Bremler-Barr, E. Cohen, and M. Shagam, "Mitigating DNS random subdomain DDoS attacks by distinct heavy hitters sketches," in *Proc. 5th ACM/IEEE Workshop Hot Topics Web Syst. Technol.*, 2017, pp. 1–6.
[40] A. Benmoussa et al., "A novel congestion-aware interest flooding attacks detection mechanism in named data networking," in *Proc. 28th Int. Conf. Comput. Commun. Netw.*, 2019, pp. 1–6.
[41] T. Zhi, H. Luo, and Y. Liu, "A gini impurity-based interest flooding attack defence mechanism in NDN," *IEEE Commun. Lett.*, vol. 22, no. 3, pp. 538–541, Mar. 2018.
[42] R. Hou et al., "Theil-based countermeasure against interest flooding attacks for named data networks," *IEEE Netw.*, vol. 33, no. 3, pp. 116–121, May/Jun. 2019.
[43] J. Chen, G. Xing, M. Cui, H. Huo, and R. Hou, "Isolation forest based interest flooding attack detection mechanism in NDN," in *Proc. 2nd Int. Conf. Hot Inf.-Centric Netw.*, 2019, pp. 58–62.
[44] J. Dong et al., "InterestFence: Countering interest flooding attacks by using hash-based security labels," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, 2018, pp. 527–537.
[45] J. Dong, K. Wang, W. Quan, and H. Yin, "InterestFence: Simple but efficient way to counter interest flooding attack," *Comput. Secur.*, vol. 88, 2020, Art. no. 101628.
[46] M. Wahlisch, T. C. Schmidt, and M. Vahlenkamp, "Backscatter from the data plane - threats to stability and security in information-centric network infrastructure," *Comput. Netw.*, vol. 57, no. 16, pp. 3192–3206, 2013.
[47] L. Yao, Y. Zeng, X. Wang, A. Chen, and G. Wu, "Detection and defense of cache pollution based on popularity prediction in named data networking," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 6, pp. 2848–2860, Nov./Dec. 2021.



Zhiwei Xu (Member, IEEE) received the BS degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2002, and the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2018. He is an associate professor and MS supervisor of Inner Mongolia University of Technology, while working as an adjunct professor with the Institute of computing, Chinese Academy of Sciences. He is currently working towards visiting post-doctoral with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, New York. His research interests include network performance analysis and the related mathematics problems.

Xin Wang (Member, IEEE) received the BS and MS degrees in telecommunications engineering and wireless communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, and the PhD degree in electrical and computer engineering from Columbia University, New York, NY. She is currently an associate professor with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, New York. Before joining Stony Brook, she was a member of technical staff in the area of mobile and wireless networking with Bell Labs Research, Lucent Technologies, New Jersey, and an assistant professor with the Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, New York. Her research interests include algorithm and protocol design in wireless networks and communications, mobile and distributed computing, as well as networked sensing and detection. She has served in executive committee and technical committee of numerous conferences and funding review panels, and serves as the associate editor of the *IEEE Transactions on Mobile Computing*. She achieved the NSF career award in 2005, and ONR challenge award in 2010.

Yujun Zhang (Member, IEEE) received the BS degree from Nankai University, Tianjin, China, in 1999, and the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2004. He is a professor and PhD supervisor with the Institute of Computing Technology, Chinese Academy of Sciences. His main research interests includes future Internet, Internet security assessment, and verification.