Needle in a Haystack:
Mitigating Content Poisoning in Named-Data Networking

Cesar Ghali
University of California, Irvine
cghali@uci.edu

Gene Tsudik
University of California, Irvine
gts@ics.uci.edu

Ersin Uzun
Palo Alto Research Center
Ersin.Uzun@parc.com

Abstract—**Named-Data Networking (NDN) is a candidate next-generation Internet architecture designed to address some limitations of the current IP-based Internet. NDN uses the pull model for content distribution, whereby content is first explicitly requested before being delivered. Efficiency is obtained via router-based aggregation of closely spaced requests for popular content and content caching in routers. Although it reduces latency and increases bandwidth utilization, router caching makes the network susceptible to new cache-centric attacks, such as content poisoning. In this paper, we propose a ranking algorithm for cached content that allows routers to distinguish good and (likely) bad content. This ranking is based on statistics collected from consumers' actions following delivery of content objects. Experimental results support our assertion that the proposed ranking algorithm can effectively mitigate content poisoning attacks.**

I. Introduction

Since its early days in the 1970-s, the way the Internet is used has changed drastically. From a few thousand people, mainly in North America, using it to access shared computing resources from terminals, Internet's usage gradually shifted to billions of people world-wide running a wide range of applications from a variety of end-devices. This shift also exposed limitations of the current IP-based Internet and motivated exploring new architectures.

Named-Data Networking (NDN) [1] is one of several research efforts aiming to design a next-generation Internet architecture. NDN is an example of Information-Centric Networking (ICN) [2]. It is based on PARC's Content-Centric Networking project [3], [4]. Unlike traditional IP-based networking that assigns addresses to hosts and interfaces, NDN addresses content by giving it a routable name. A consumer issues an *interest* for a certain content, requesting it by name. The network can satisfy an interest with a *content* from any host or router cache, as long as content name matches that in the interest. An NDN content follows, in reverse, the exact path of the preceding interest, all the way back to the *consumer*.

To ensure authenticity, NDN stipulates that every content must be signed by its producer and consumers are required to verify content signatures. However, content signature verification is optional for routers, due mainly to various associated costs.

To facilitate efficient distribution of popular content, NDN routers maintain so-called Content Stores, where content is cached. This caching is a key NDN feature as it reduces overall latency and improves bandwidth utilization for popular content. However, despite its obvious benefits, content caching in routers opens the door for DoS attacks [5].[1] One such attack is content poisoning whereby an adversary injects fake content into router caches thus resulting in its subsequent (possibly, large-scale) distribution to consumers.

Although signature verification by consumers would detect invalid content, NDN has no means of facilitating its removal from router caches, other than eventual natural (e.g., LRU-based) cache aging. The only way a consumer can attempt to avoid re-receiving fake content is via explicit exclusion filters (containing one or more hashes of not-desired content) in NDN interest packets.

In this paper, we propose a light-weight ranking algorithm for cached content in NDN routers to mitigate content poisoning. Our approach is based on statistics collected from existing fields of interest packets and does not require any changes to the NDN architecture. Simulation-based experiments show that the ranking algorithm is effective against content poisoning attacks under several adversarial assumptions.

Although vulnerability of NDN architecture to content poisoning attacks was noted recently [5], to the best of our knowledge, no prior research explores this problem in any depth or proposes any concrete counter-measures. This paper focuses on c ontent-poisoning attacks in NDN by: (1) showing their potential via experiments, and (2) constructing and evaluating a simple and light-weight content ranking technique. Anticipated contributions are as follows:

- First study of NDN's vulnerability to content poisoning attacks and their impact on quality of service.
- First steps towards a concrete and comprehensive solution against content poisoning attacks in NDN via a light-weight statistical content-ranking algorithm for router caches.
- Evaluation of the proposed technique under a variety of topologies and cache contamination assumptions.

SENT '14, 23 February 2014, San Diego, CA, USA
Copyright 2014 Internet Society, ISBN 1-891562-36-3
http://dx.doi.org/10.14722/sent.2014.23014

[1] It also facilitates attacks on consumer and producer privacy, as discussed in [6], [7], [8].

NDN is one of five Future Internet Architecture projects funded by the U.S. National Science Foundations (NSF) [9]. NDN is thus a contender for the next-generation Internet architecture. Even if it fails to win the proverbial "race", its design might influence the Internet of the future, i.e., certain NDN features – such as router caching – could be adopted. Thus, we believe that this paper is both timely and important, since it studies NDN's vulnerability to a particular type of DoS attack (which stems from router caching) and proposes a simple yet efficient and effective counter-measure.

Organization: Section II describes the NDN architecture. Section III presents the adversary model, definitions and the terminology. Then, Section IV describes the content ranking algorithm. Section V discusses our experimental environment. Next, Section VI overviews related work and Section VII concludes the paper.

II. NDN Overview

This section overviews NDN. Given some familiarity with NDN, it can be skipped with no loss of continuity.

Unlike IP which focuses on end-points of communication and their names/addresses, NDN ([3], [1]) focuses on content and makes it named, addressable and routable at the network layer. A content name is composed of one or more variable-length components opaque to the network. Component boundaries are explicitly delimited by "/" in the usual path-like representation. For example, the name of a CNN news homepage content for May 20, 2013 might be: `/ndn/cnn/news/2013may20/index.htm`. Large content can be split into segments with different names where fragment 37 of Alice's YouTube video could be named: `/ndn/youtube/alice/video-749.avi/37`.

NDN communication adheres to the *pull* model and content is delivered to consumers only upon explicit request. There are two types of packets in NDN: interest and content. A consumer requests content by issuing an *interest* packet. If an entity can "satisfy" a given interest, it returns a corresponding *content* packet. Each content delivery in NDN must be strictly preceded by an interest. If content C with name n is received by a router with no pending interest for that name, the content is considered unsolicited and is discarded. Name matching in NDN is prefix-based. For example, an interest for `/ndn/youtube/alice/video-749.avi` can be satisfied by content named `/ndn/youtube/alice/video-749.avi/37`.[2]

<u>REMARK:</u> we use these three terms: *content object*, *content packet* and *content* interchangeably.

NDN content packets include several fields. In this paper, we are only interested in three:

- `Name` – A sequence of explicit name components followed by an implicit digest (hash) component of the content recomputed at every hop. This effectively provides each content with a unique name and guarantees a match when provided in an interest. However, in most cases, the hash component is not present in interest packets, since NDN does not provide any secure mechanism to learn content hashes a priori.
- `Signature` – a public key signature, generated by the content producer, covering the entire object, including all explicit components of the name. The signature field also includes a reference (by name) to the public key needed to verify it.
- `Freshness` – producer-recommended time for the content objects to be cached.

Each producer is required to have at least one public key, represented as a *bona fide* named content object, signed by its issuer, e.g., a CA. (NDN is fully agnostic as far as trust management, allowing both peer-based and hierarchical PKI approaches.) The name of a public key content object must contain "key" as its last explicit component, e.g., `/ndn/russia/moscow-airport/transit/snowden/key`. Moreover, in order for content signature to be valid (not just verifiable), the name of the public key (the private counterpart of which is used to sign) without the last explicit component must form a prefix of the content name. For example, this would hold for content named `/ndn/russia/moscow-airport/transit/snowden/rant` and key named as above, but would not hold for key named: `/ndn/usa/nsa/leaker/snowden/key`.

NDN interests packets include the following fields:

- `Name` – Name of the requested content.
- `MinSuffixComponents` and `MaxSuffixComponents` – respectively, minimum and maximum number of name components, beyond those specified in the name, that are allowed to occur in matching content. These fields facilitate longest-prefix matching.
- `Exclude` – contains information about name components that **must not** occur in the name of returned content. This field can be used to exclude certain content by referring to its hash, which, as noted above, is considered to be an implicit, last component of each content name.

There are three types of NDN entities/roles:[3]

- *Consumer* – an entity that issues an interest for content.
- *Producer* – an entity that produces and publishes (as well as signs) content.
- *Router* – an entity that routes interest packets and forwards corresponding content packets.

Each NDN entity (not just routers) maintains these three components [10]:

- *Content Store* (CS) – cache used for content caching and retrieval. From here on, we use the terms *CS* and *cache* interchangeably. Recall that timeout of cached content is specified in the freshness field.
- *Forwarding Interest Base* (FIB) – table of name prefixes and corresponding outgoing interfaces. FIB is used to route interests.
- *Pending Interest Table* (PIT) – table of outstanding (pending) interests and a set of corresponding incoming and outgoing interfaces.

When a router receives an interest for name n, and there are no pending interests for the same name in its PIT, it

[2] However, the reverse does not hold, by design.

[3] A physical entity (a host, in today's parlance) can be both consumer and producer of content.

forwards the interest to the next hop(s), according to its FIB. For each forwarded interest, a router stores some amount of state information, including the name in the interest and the interface on which it arrived. However, if an interest for n arrives while there is already an entry for the same content name in the PIT, the router collapses the present interest (and any subsequent interests for n) storing only the interface on which it was received. When content is returned, the router forwards it out on all incoming-interest interfaces and flushes the corresponding PIT entry. Since no additional information is needed to deliver content, an interest does not carry any *source address*.

A router's cache size is determined by local resource availability. Each router unilaterally determines what content to cache and for how long. Upon receiving an interest, a router first checks its cache to see if it can satisfy this interest locally. Therefore, NDN lacks any notion of a *destination address* – content can be served by any NDN entity. Producer-originated content signatures allow consumers to authenticate received content, regardless of the entity that serves this content. As mentioned above, [11] consumers must verify content signatures. However, signature verification is optional (and viewed as unlikely) for routers, since doing so can be prohibitively expensive at line speed.

III. Content Poisoning

This section describes content poisoning attack scenarios. We begin with the terminology used in the rest of the paper.

- A **fake** content has one of the following features:
  - ○ An invalid signature, i.e., the signature verification algorithm returns an error.
  - ○ A valid signature which is generated (signed) with the wrong key, i.e., not the key of the purported producer.
  - ○ A signature field that is somehow mal-formed, e.g., formatted badly.

  From here on, we use **fake** to refer to content objects injected by the adversary.
- A content object is **valid** if it contains a verifiable signature produced with the correct public key. (The meaning of *correct* public key is discussed in Section II above.)
- **Adversary**, Adv, is any NDN entity (or a collaborating group thereof) capable of injecting content into the network.
- **Content poisoning** is an attack whereby Adv injects fake content into router caches.

In this paper, we consider a pro-active content poisoning attack whereby Adv anticipates interests for content C with name n and injects fake content with the same name into router caches. Fake content can be injected into the network via malicious routers or end-nodes. For example, consider an Adv (consisting of malicious consumer Cr_m and a malicious producer P_m) targeting a specific router R_v. Assuming that Cr_m and P_m are connected to different interfaces of R_v, Cr_m sends an interest for n. Once this interest is received by R_v and an entry is added to the PIT, P_m sends a fake content to R_v which is promptly cached. Consequently, R_v is pre-polluted with fake content, ready for arrival of genuine interests. To maximize longevity of the attack, we assume P_m sets the freshness field of fake content to a maximum value.

IV. Content Ranking

In principle, preventing content poisoning is not particularly difficult. If every router verifies the signature of each content object prior to serving and caching it [11], the problem simply goes away. However, modern routers cannot perform signature verification (even using the most efficient techniques) at line speed. Also, verifying signatures requires a mechanism to fetching, parsing and verifying public keys. However, since trust is always application-dependent, involving (especially backbone) routers in the specifics of trust management is patently absurd.

Another way to mitigate content poisoning is by enforcing the use of self-certifying content names. This is easy to do since content can always be referred by its full name (where the last component is the hash of the very same content) and setting the `MaxSuffixComponents` field to zero. However, this triggers the chicken-and-egg situation: how does the consumer obtain the content hash in the first place? One partial answer is that hashes can be distributed via so-called catalogs. A catalog is a type of content that includes names and corresponding hashes of other content, e.g., signed content representing the main page of a (static) web-site might include hashes of all sub-pages. Thus, a consumer first obtains a catalog and uses it to construct self-certifying names for subordinate content. Nonetheless, how a consumer can request the catalog content, since it does not know its hash in advance. Another problem with self-certifying names arises in the context of dynamically-generated content, since its hash cannot be created before the content itself.

Motivated by the above discussion, we explore lighter-weight approaches for mitigating content-poisoning. In particular, we present a ranking algorithm for cached content objects. Its objective is to probabilistically distinguish between valid and fake content objects based on observed consumer behavior and prioritize valid over fake content in response to consumer interests. The proposed algorithm is fully compatible with the current NDN design – it requires no architectural modifications or coordination between routers.

Our approach is premised on the fact that consumers who verify signatures and detect fake content subsequently issue a new interest that excludes the previously received fake content. Our hypothesis is that analyzing exclusion information could allow routers to rank cached content such that valid content is ranked higher than fake content. We achieve this by assigning each cached content object a rank and have routers select the highest-ranked object in response to an interest. The rank is a numeric value between 0 and 1, where 1 is highest. All cached content starts with the rank of 1, and, as time elapsed, this value gradually decreases. This gives priority to newer cached content over older. In addition, the rank of a specific content depends on the number of times it was excluded and when. We assign a lower rank to content with many recent exclusions, over fewer old ones. The reasoning is that few exclusions might always occur normally because even valid content objects might not always satisfy a given consumer interest. Also, a consumer might exclude certain content since it represents a wrong (e.g., old/stale) version. Finally, we harshly penalize content which has been excluded by interests arriving on multiple interfaces, since that would indicate higher likelihood of something being wrong with the content in question.

A. *Number of Exclusions*

This criterion is based on the number of times content object C with name n has been excluded. Different versions of C might have the same name and different data thus resulting in different hashes. To distinguish among such identically-named content, we denote each distinct version as $n|H(C)$. The exclusion rate is the ratio of the number of exclusions for $n|H(C)$, denoted as $E_{n|H(C)}$ and the total number of requests for C, Q_n. It is denoted as $R_{n|H(C)} = E_{n|H(C)}/Q_n$. We consider the ratio as more meaningful than the sheer number of exclusions. The reason is that consumers can exclude not only fake, but also incorrect[4] content objects. Our hypothesis is that fake content objects tend to have much higher exclusion rates.

Content objects ranking degradation pattern can be modeled by Equation 1.

$$r_{n|H(C)}(t) = e^{\frac{-t}{\alpha}} \tag{1}$$

where t is the age of $n|H(C)$ ($t \in [0, t_{to}]$; t_{to} is content freshness) in the cache, and α is a factor determining the degradation speed of the content rate. Note that $r_{n|H(C)} \in [r_{n|H(C)}(t_{to}), 1]$. The larger the value of α, the faster the content's rank degrades.

We still need to define α. Let r_{to} be the rank of cached $n|H(C)$ when it expires (freshness time elapsed), provided that $n|H(C)$ is never excluded during its lifetime. r_{to} is a system parameter that can be determined by the network administrator. For a given freshness value and a desired value of r_{to}, the corresponding value of α can be computed from Equation 1. We denote this value as α_{to} – the value of α that makes the rank of a non-excluded content object r_{to}, when it expires. We want to assign a higher rank to objects excluded less. In other words, the more exclusions for $n|H(C)$, the sharper its degradation factor. If $n|H(C)$ is never excluded, the rank degradation pattern is the slowest from 1 to r_{to}.

Based on the above discussion, α is dependent on α_{to} and negatively affected by $R_{n|H(C)}$, $\alpha = \alpha_{to} - \big(R_{n|H(C)} \times \alpha_{to}\big)$. Therefore, Equation 1 can be rewritten as follows:

$$r_{n|H(C)}(t) = e^{\frac{-t}{\alpha_{to} - \left(R_{n|H(C)} \times \alpha_{to}\right)}} \tag{2}$$

B. *Time Distribution of Exclusions*

We now factor into the ranking algorithm the timing of exclusions. The main idea is to give more weight to newer exclusion. We define exclusion influence $i_{n|H(C)}$ as the variable reflecting the time that the router should wait before the effect of the latest exclusion attempt for $n|H(C)$ is assigned the minimal weight when the rank is calculated. This is computed as:

$$i_{n|H(C)}(t_e) = 1 - e^{\frac{-t_e}{\beta}} \tag{3}$$

where t_e is time elapsed since the last exclusion and β is a factor determining the influence pattern; it reflects how fast the effect of the latest exclusion is minimally weighted. Note that $i_{n|H(C)}(t_e) \in [0, 1]$, where $i_{n|H(C)}(t_e) = 1$ means that the latest exclusion has minimal effect on ranking.

The larger β, the more time should elapse before minimally weighting the latest exclusion. This time is denoted as t_{mw}. Given a value of t_{mw}, which can be set by the network administrator, and setting $i_{n|H(C)}(t_e) = 1$, the corresponding β can be computed from Equation 3. We denote it as β_{mw}.

We can modify Equation 2 to include the exclusion influence factor as follows:

$$r_{n|H(C)}(t) = e^{\frac{-t}{i_{n|H(C)}(t_e) \times \left[\alpha_{to} - \left(R_{n|H(C)} \times \alpha_{to}\right)\right]}} \tag{4}$$

If $i_{n|H(C)}(t_e) = 1$, the rank of the $n|H(C)$ is only affected by exclusion rate $R_{n|H(C)}$.

C. *Excluding Interfaces Ratio*

This criterion considers the number of interfaces on which exclusions arrive. A higher number indicates higher dissatisfaction with specific content. This can be used to penalize ranking of that content. Consider the following:

- f_n – total # of interfaces of a given router.
- $f_e \in [0, f_n]$ – # of interfaces on which the router received interests excluding $n|H(C)$.
- $f_s \in [1, f_n]$ – # of interfaces on which the router previously served $n|H(C)$. (Note that f_s can not be zero, since for ranking to exist, the corresponding content must have been requested and served on at least one interface).
- $e_{n|H(C)}$ – ratio of the number of interfaces on which the router previously served $n|H(C)$ but no exclusion arrived to f_s.

$$e_{n|H(C)} = \begin{cases} \frac{f_s - f_e}{f_s} & \text{if } f_s \geq f_e \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

$e_{n|H(C)} \in [0, 1]$ where 1 means that $n|H(C)$ was not excluded at all. However, note that f_e might actually exceed f_s. More generally, it is possible for a router to receive an interest excluding $n|H(C)$ on an interface where this content has not been served. This could occur for at least three benign reasons: (1) routing changes, (2) consumer mobility, and (3) cache replacement. The first two are self-explanatory, whereas (3) refers to the case when content was previously requested, served, cached, and then flushed from the cache, for any reason, including normal aging.

Based on the previous definition we can rewrite equation 4 as follows:

$$r_{n|H(C)}(t) = e^{\frac{-t}{e_{n|H(C)} \times i_{n|H(C)}(t_e) \times \left[\alpha_{to} - \left(R_{n|H(C)} \times \alpha_{to}\right)\right]}} \tag{6}$$

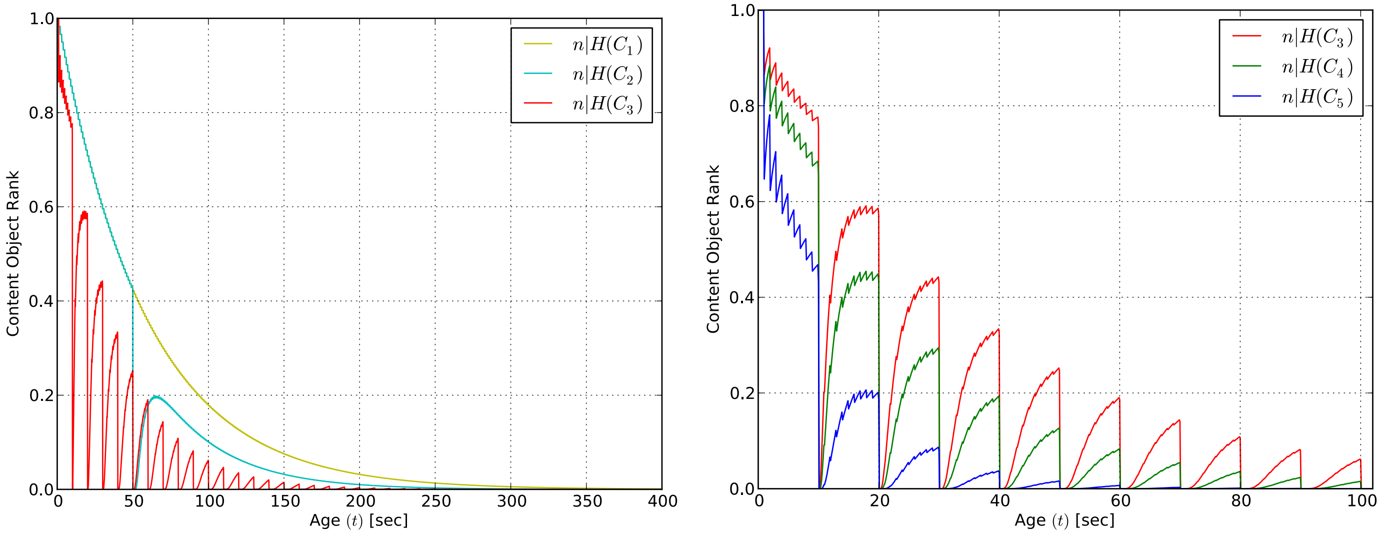
D. *Analysis*

Equation 6 reflects the rank of each content object cached by any router. This ranking is based on three criteria: the number of exclusions, the time distribution of the exclusion attempts, and the excluding interfaces ratio. Next, we show a comparison between ranking degradation patterns of five cached content objects. Parameters of Equation 6 differ for

[4] An *incorrect* content object is an otherwise valid content object that does not satisfy the consumer's interest.

TABLE I. Content Objects Parameters

Parameter	$n\|H(C_1)$	$n\|H(C_2)$	$n\|H(C_3)$	$n\|H(C_4)$	$n\|H(C_5)$
Content	C_1	C_2	C_3	C_4	C_5
Name	n	n	n	n	n
Digest	$H(C_1)$	$H(C_2)$	$H(C_3)$	$H(C_4)$	$H(C_5)$
t	$[0, 400]$, one sample every 100 $[msec]$				
freshness	400	400	400	400	400
r_{to}	0.001	0.001	0.001	0.001	0.001
Q_n	1	1 when $t \in [0, 50]$, and 2 when $t \in (50, 400]$	increased by one every 10 $[sec]$		
$E_{n\|H(C)}$	0	0 when $t \in [0, 50]$, and 1 when $t \in (50, 400]$	increased by one every 10 $[sec]$		
t_{mw}	400	400	400	400	400
t_e	∞	∞ when $t \in [0, 50]$, and increased by one every 1 $[sec]$ when $t > 50$	$[0, 10]$, increased by one every 1 $[sec]$, and reset every 10 $[sec]$		
f_n	4	4	4	4	4
f_e	0	0 when $t \in [0, 50]$ 1 when $t \in (50, 400]$	1	2	3



(a) $n|H(C_1)$: never excluded, $n|H(C_2)$: excluded once when $t = 50$ seconds, and $n|H(C_3)$: excluded every 10 seconds

(b) $n|H(C_3)$: excluded on 1 interface, $n|H(C_4)$: excluded on 2 interface, and $n|H(C_5)$: excluded on 3 interface. All three objects are excluded every 10 seconds.

Fig. 1. Content object ranking comparison

each object; they are summarized in Table I. These content objects are:

- $n|H(C_1)$: requested only once and never excluded throughout its lifetime in the router's cache.
- $n|H(C_2)$: requested only once, and at time $t = 50$ seconds, excluded once.
- $n|H(C_3)$, $n|H(C_4)$, and $n|H(C_5)$: requested once without exclusion and then excluded every 10 seconds. The difference between these three objects is their excluding interfaces ratio.

Based on Equation 6 and the above five contents, we assume that $n|H(C_1)$ should have higher ranking at all times, followed by $n|H(C_2)$, then $n|H(C_3)$, $n|H(C_4)$, and $n|H(C_5)$.

Figure 1 confirms our assumptions by showing ranking degradation patterns of five content objects. Figure 1(a) shows that, when $t <= 50$ seconds, both $n|H(C_1)$ and $n|H(C_2)$ have equal ranking values, which is higher than the other three excluded content objects. The ranking of $n|H(C_2)$ decreases after it is excluded at $t = 50$ seconds. Moreover, the repetitive pattern of $n|H(C_3)$ is justified because this content object is excluded every 10 seconds. Once an exclusion occurs, ranking drops to close to 0, and starts increasing again according to Equation 3. On the other hand, Figure 1(b) compares between $n|H(C_3)$, $n|H(C_4)$, and $n|H(C_5)$ in a shorter time window of 100 seconds. It demonstrates the effect of varying f_e. For instance, $n|H(C_5)$ is excluded on 3 different interfaces, while $n|H(C_3)$ is excluded on only one. Thus, the former has a lower ranking value.

Based on our definitions and the analysis of the content ranking algorithm, we conclude that newer content objects have higher ranking than old ones. This is an intentional design feature to give newer content priority in distribution and chance for timely dissemination. Moreover, in cases of none or few malicious consumers, newer content objects are less likely to be fake. This is due to the fact that a router always tries to satisfy an interest from its cache. As long as a router's cache contains a valid version of content, consumers are served that content and are unlikely to send another interest that excludes valid content. Therefore, the only common case where a fake content can be cached later than a valid one is when malicious consumers downstream work against the ranking algorithm by

TABLE II. NDNSIM TOPOLOGIES PARAMETERS

Parameter	**Tree-based Topology**	**DFN Topology**		**AT&T Topology**	
# of consumers	50	80	80	160	160
# of routers	6	30	30	132	132
# of producers	0	0	0	0	0
Cache replacement policy	LRU	LRU	LRU	LRU	LRU
Simulation time [sec]	400	400	400	400	400
Pre-populated fale content objects rate	99.9%	80%, 90%, 99%, and 99.9%	99%, and 99.9%	80%, 90%, 99%, and 99.9%	99%, and 99.9%
Pre-populated content objects freshness [sec]	400	400	400	400	400
Malicious consumer rate	0%, 2%, 4% 6%, and 10%	0%	0%, 1%, 3%, 5%, and 10%	0%	0%, 1%, 3% 5%, and 10%
Interest interval [millisecond]	[100, 300]	[100, 300]	[100, 300]	[100, 300]	[100, 300]

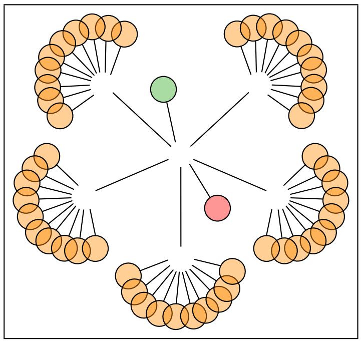


Fig. 2. Tree-based topology - orange: consumer, blue: router, green: producer, red: Adv

excluding valid content, or, explicitly request fake content. More details on this case are in Section V. Our approach remains effective even with powerful distributed attackers as long as benign consumers are not outnumbered by malicious ones.

V. NDNSIM EXPERIMENTS

ndnSIM [12] is a simplified implementation of NDN architecture as a NS-3 [13] module for simulation purposes. To verify correctness and practicality of the proposed content ranking algorithm, we extended ndnSIM to incorporate our method in router caches. The rest of this section describes our simulation scenarios and experiments, followed by the discussion of results. The following terminology is used:

- **Benign** consumers are not satisfied if an interest returns a fake content object. After receiving a fake content, they always exclude it in their subsequent interests for the same name.[5] Moreover, benign consumers stop sending interest messages after receiving valid content.
- **Malicious** consumers behave in the opposite manner. If an interest returns a valid content, they exclude it in all subsequent interests for the same name. The objective of a malicious consumer is to change statistics collected about exclusions to favor fake content.

We use simulations to measure how many benign consumers can retrieve a satisfactory (genuine) content and how fast they can do so when the router caches are poisoned. The simulation starts with router caches pre-populated with various fake versions of the target content. We vary the rate of fake pre-populated content, as discussed below. Table II shows the parameters for our experiments.

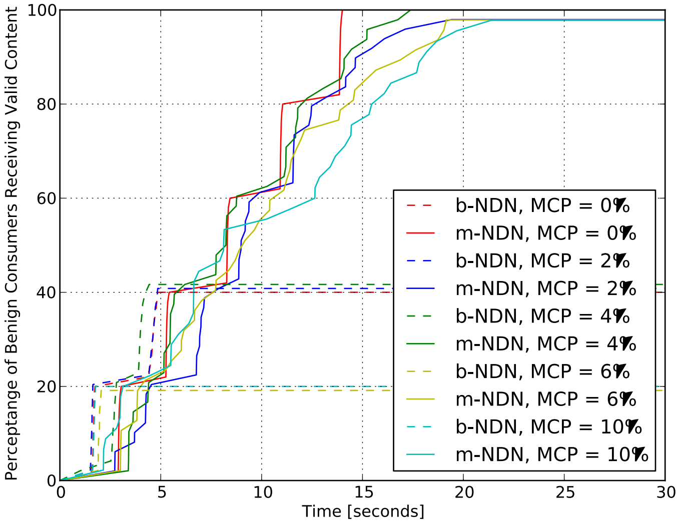


Fig. 3. Tree-based topology with various malicious consumer rates (b-NDN: basic NDN with LRU cache replacement, m-NDN: modified NDN with routers implementing our ranking algorithm, MCP: percentage of malicious nodes in the consumer population)

A. *Tree-based Topology*

Tree-based topology is illustrated in Figure 2. It consists of 5 consumer-facing routers (each connected to 10 consumers) connected to a single backbone router. The topology also contains one producer and one adversary[6], both connected to the backbone router. Such a topology is commonly formed in distributing content from a single producer in ICNs.

In our experiments, we first pre-populate the network routers with 1000 different versions of the same content, only one of them is valid, i.e. the pre-populated fake content objects rate is 99.9%. In addition, different rates of the 50 consumers

[5] In our implementation, the maximum number of hashes that can be included in an interest is 100. When that number is reached, a new hash added to the exclusion list replaces the oldest one.

[6] The adversary connected to the same backbone router with the producer is to demonstrate how a strategically located adversary can easily distribute fake content into caches. However, we skip the initial phase of the attack and start with caches that are pre-populated with fake content.

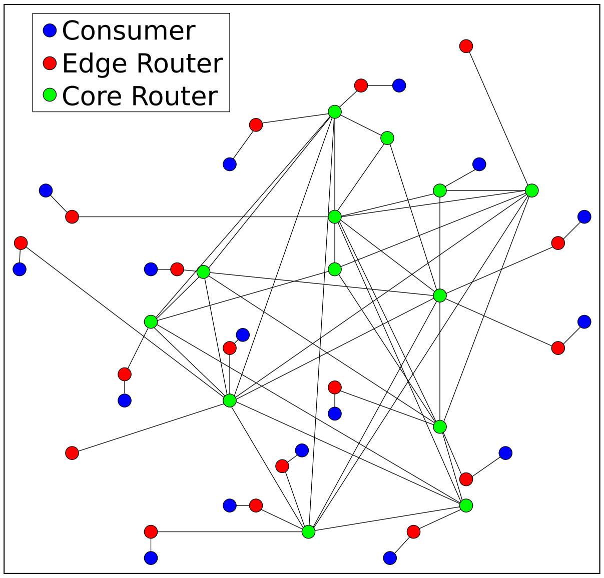


Fig. 4. DFN topology - each edge router above is connected to 5 NDN consumers

are configured as malicious, 0%, 2%, 4%, 6%, and 10%. Figure 3 shows the results of this experiment. We can notice that in the cases where ranking is not applied the network was not able to reach a state where all benign consumers receive valid content. When this event occurs, we say that the network is converged to a state in which all benign consumers have stopped. We call this state a full convergence. The reason why the network is not able to fully converge is because consumers can only exclude up to certain number of fake content objects, which is 100 in our configuration, in interest messages. However, for all different malicious consumer rates simulated, applying ranking always leads the network to full convergence, even though it takes longer for higher malicious consumers rates.

B. DFN Topology

After verifying the correct behavior of our ranking algorithm using the tree-based topology, we consider a more complex network topology. The DFN network, Deutsches ForschungsNetz (German Research Network) [14], [15], is a German network developed for research and education purposes. It consists of several connected routers positioned in different areas of the country, as shown in Figure 4. The network consists of a total of 30 routers and 80 consumers.

We ran two sets of experiments using the DFN topology.

1) All routers in the network are pre-populated with different rates of fake content objects, 80% (1 valid and 4 fake content objects), 90% (1 valid 9 fake objects), 99% (1 valid 99 fake objects), and 99.9% (1 valid 999 fake objects). Figure 5 shows the results of this experiment. We can notice that the network reaches full convergence in all the cases except when ranking is not applied for pre-populated fake content objects rate of 99.9%. The reason is similar to what explained before regarding the size of exclusion filters.
2) In this experiment, we vary the malicious consumers rate (0%, 1%, 3%, 5%, and 10%) for two rates of pre-populated fake content objects, 99% and 99.9%. Figures 6 and 7 show the obtained results respectively. In Figure 6, we can notice that the ranking algorithm allows faster full convergence, while in Figure 7, the network is not able to fully converge without applying our ranking algorithm.

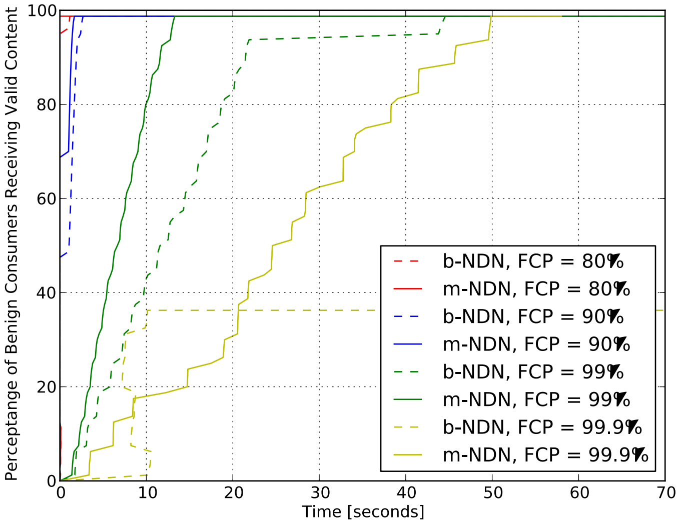


Fig. 5. DFN topology results with different rates of pre-populated fake content objects (b-NDN: basic NDN with LRU cache replacement, m-NDN: modified NDN with routers implementing our ranking algorithm, FCP: percentage of pre-populated fake content objects)

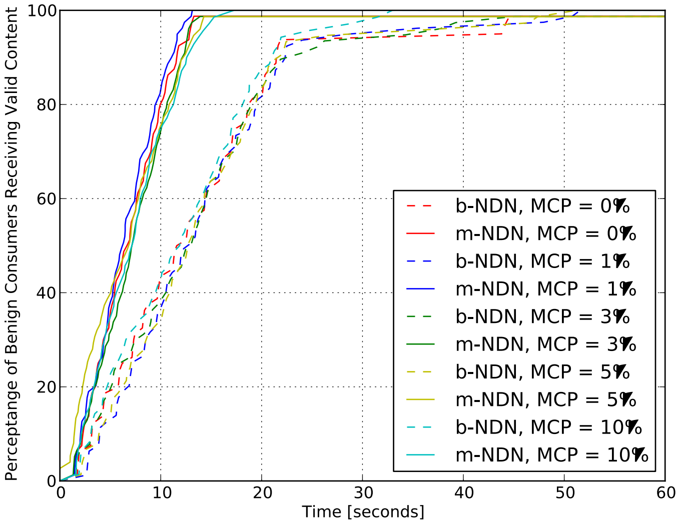


Fig. 6. DFN topology results with different rates of malicious consumers and 99% pre-populated fake content objects (b-NDN: basic NDN with LRU cache replacement, m-NDN: modified NDN with routers implementing our ranking algorithm, MCP: percentage of malicious nodes in the consumer population)

C. AT&T Topology

In this section, we evaluate the performance of our ranking algorithm in a much bigger topology, the AT&T backbone network [16] shown in Figure 8. This network consists of more than 130 routers and a total of 160 consumers.

We evaluate the performance of our ranking algorithm using the following two sets of experiments. The analysis of the obtained results is similar to that in Section V-B.

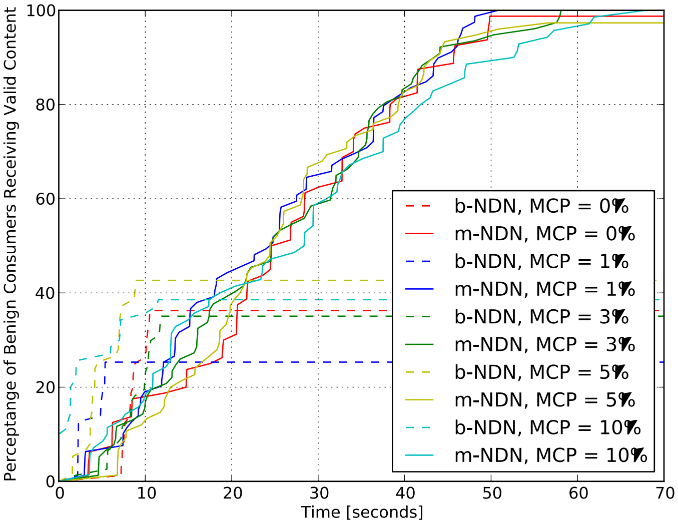


Fig. 7. DFN topology results with different rates of malicious consumers and 99.9% pre-populated fake content objects (b-NDN: basic NDN with LRU cache replacement, m-NDN: modified NDN with routers implementing our ranking algorithm, MCP: percentage of malicious nodes in the consumer population)

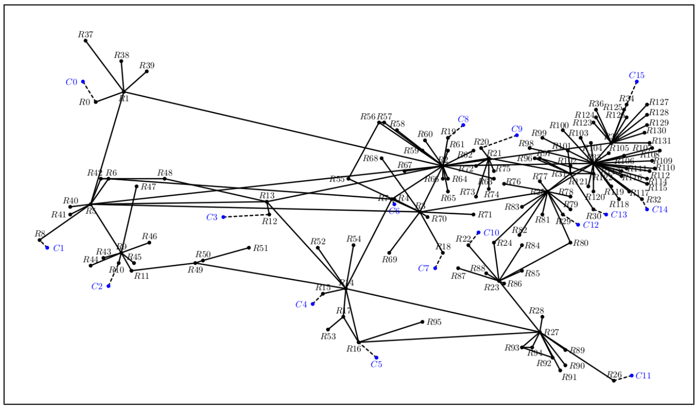


Fig. 8. AT&T topology - each edge router above is connected to 10 NDN consumers

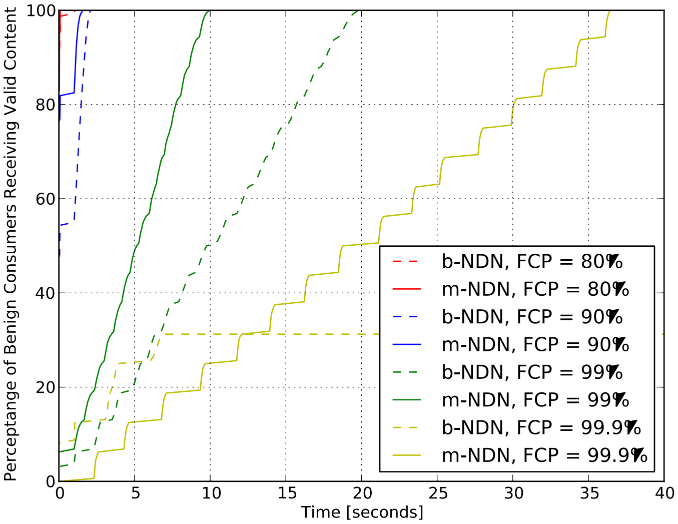


Fig. 9. AT&T topology results with different rates of pre-populated fake content objects (b-NDN: basic NDN with LRU cache replacement, m-NDN: modified NDN with routers implementing our ranking algorithm, FCP: percentage of pre-populated fake content objects)

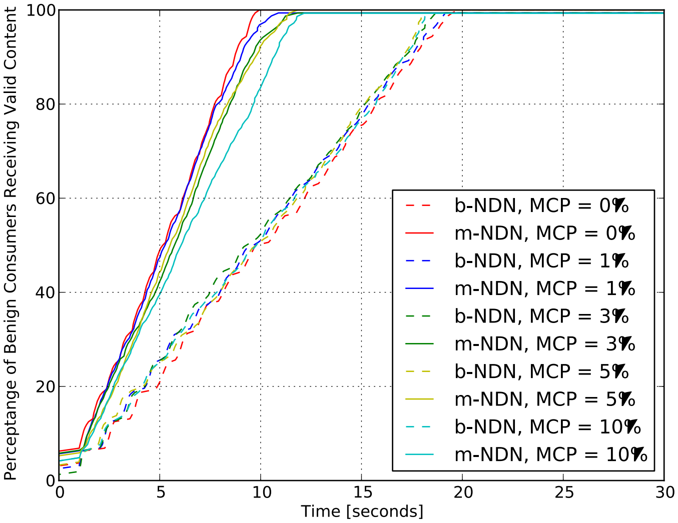


Fig. 10. AT&T topology results with different rates of malicious consumers and 99% pre-populated content objects (b-NDN: basic NDN with LRU cache replacement, m-NDN: modified NDN with routers implementing our ranking algorithm, MCP: percentage of malicious nodes in the consumer population)

1) We pre-populate the network routers with different ratios of valid to fake content objects, 80%, 90%, 99%, and 99.9%, and we run the experiments with and without ranking. Figure 9 shows the obtained results.
2) All network routers are pre-populated with either 99% or 99.9% fake content object rates. Figures 10 and 11 show the results of running this experiment for various malicious consumer rates (0%, 1%, 3%, 5%, and 10%).

Results from simulations on the AT&T network topology show that the proposed content ranking algorithm improves the quality of service and the resilience of the network against content-poisoning attacks, even if caches of all routers in a relatively large topology are poisoned.

D. Performance Analysis

As mentioned before, we implemented a proof of concept model for our proposed content ranking algorithm using ndnSIM. To facilitate content lookup, cache can be implemented as a hash table with content name, except the last name component (the digest), serving as the key. Each key points to a priority queue [17] where the first element contains the content object with the highest rank. In the case when interest does not include exclusion filter, content lookup requires $\mathcal{O}(1)$ operations. However, when an exclusion filter is provided, lookup takes up to $\mathcal{O}(k \log n)$ operations (if implemented using maximum heap [18]), for n is the number of content objects in the priority queue and k is the number of excluded content in the interest message. The reason is because the rank of each excluded content objects needs to be recalculated and the priority queue needs to be rearranged. In terms of storage consumption, less than 50 extra bytes per cache entry is required to store all the parameters needed to calculate the rank.

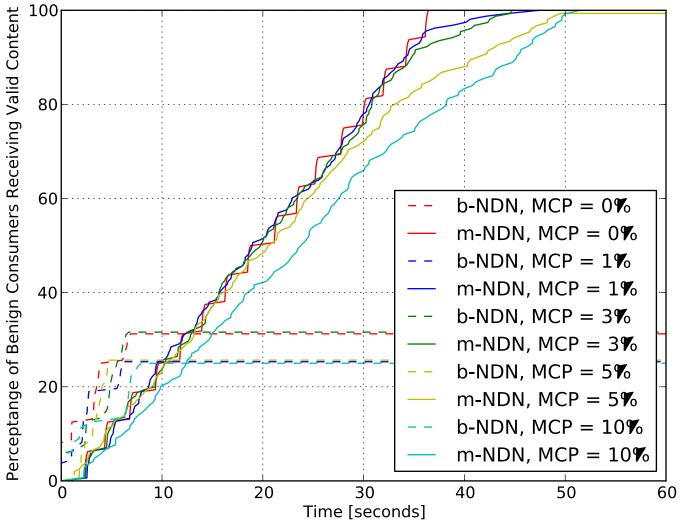


Fig. 11. AT&T topology results with different rates of malicious consumers and 99.9% pre-populated content objects (b-NDN: basic NDN with LRU cache replacement, m-NDN: modified NDN with routers implementing our ranking algorithm, MCP: percentage of malicious nodes in the consumer population)

VI. Related Work

Although several research works in the literature addressed DoS attacks in NDN, i.e. [16], [19], to the best of our knowledge, this is the first study on content poisoning attacks.

Several countermeasures have been proposed in the literature as a solution for ARP poisoning. `arpwatch` [20] is a network tool that observes ARP protocol messages and keeps track of "potentially" valid MAC/IP mappings. Whenever an anomaly is detected, `arpwatch` notifies the network administrator for further actions. In addition, Snort [21], an Intrusion Detection System (IDS), follows similar procedure in detecting ARP poisoning. However, administrator incapabilities and high false positives might raise performance inefficiencies.

In [22], Nam et. al proposed a voting-based resolution for mitigating ARP poisoning attack against the network gateway. When a new node is turned on in a LAN environment, all other nodes transmit to it their MAC/IP mapping of the gateway. This allows the new node to detect any malicious mapping advertisements for the gateway. In addition, Trabelsi et. al proposed in [23] using a stateful cache and apply a fuzzy logic approach to detect MAC/IP mappings anomalies. In their model, network nodes share trust information about all other nodes. Unlike the current design of ARP, when an ARP request is sent, the sender waits to receive all replies and selects (based on the trust information) the one that presumably is the most trustworthy.

Another type of cache poisoning attacks is DNS spoofing. It is the first step in mounting a MITM attack. The adversary maliciously changes valid entries in DNS servers by exploiting found vulnerabilities in their DNS daemon, as described in [24]. As a results, legitimate users will be redirected to false destinations.

DNSSEC [25] solves the DNS cache poisoning attack (spoofing) by ensuring the authenticity of DNS responses with a digital signature. These responses are trusted if and only if the signature is successfully verified. The way how DNS works [26] provides a built-in mechanism for lower layers DNS servers public keys distribution. End users only need to trust the root servers which provides the key of the next DNS server in the chain.

Many research efforts built solutions based on DNSSEC trying to improve its performance. In [27], Sun et. al proposed deploying a new DNS client that queries multiple DNS resolvers instead of trusting only one. Bassil et. al presents in [28], S-DNS, a secure and backward compatible protocol that provides lower computation and communication overhead, compared to DNSSEC, by replacing the traditional public-key infrastructure with an efficient identity-based encryption. Perdisci et. al [29] proposed WSEC DNS that exploits the definition of wildcard domain names and the fact that all DNS responses must copy the information in the corresponding requests. Using widecards increases the entropy of DNS requests and renders guessing the response a hard task. WSEC DNS is easily deployable because of its fully backward compatibility, unlike DNSSEC.

VII. Conclusions

NDN is one of several candidates for the next-generation Internet architecture. Despite many built-in security features, it still susceptible to some new threats, such as content poisoning, whereby the adversary injects fake content into router caches. Such objects are later served to consumers in response to interest packets.

In this paper, we proposed a content ranking algorithm for detecting and mitigating content poisoning attacks in NDN. It is based on consumer actions upon receiving fake content. All NDN nodes collect statistics about excluded content objects and assign to each cached item a numerical value – a rank. Then, the highest ranked cached object is selected as a candidate for satisfying all subsequent consumers interests. Experimental results support our assertion that the proposed ranking algorithm detects and mitigates content poisoning attacks. To the best of our knowledge, this is the first research effort that addresses content poisoning in NDN. In the future, we would like to assess the performance of our ranking algorithm in the presence of an active adversaries continuously replying with fake content objects for received interest messages.

Acknowledgments

This work was funded by NSF under the "CNS-1040802: FIA: Collaborative Research: Named Data Networking (NDN)" project.

References

[1] "Named Data Networking project (NDN)," http://named-data.org.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 26–36, 2012.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*. ACM, 2009, pp. 1–12.

[4] "Content centric networking (CCNx) project," http://www.ccnx.org.

[5] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS & DDoS in named-data networking," in *Proceedings of the 22nd International Conference on Computing Communications and Networks*. IEEE, 2013.

[6] A. Chaabane, E. De Cristofaro, M. A. Kaafar, and E. Uzun, "Privacy in content-oriented networking: Threats and countermeasures," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 25–33, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2500098.2500102

[7] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun, "Andana: Anonymous named data networking application," in *NDSS*. The Internet Society, 2012.

[8] G. Acs, M. Conti, P. Gasti, C. Ghali, and G. Tsudik, "Cache privacy in named-data networking," in *Proceedings of the 33rd International Conference on Distributed Computing Systems*. IEEE, 2013.

[9] "National Science Foundation of future Internet architecture (FIA) program," http://www.nets-fia.net/.

[10] "CCNx Node Model," http://www.ccnx.org/releases/latest/doc/technical/CCNxProtocol.html.

[11] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named data networking (ndn) project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[12] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim: NDN simulator for NS-3," *University of California, Los Angeles, Technical Report*, 2012.

[13] "Network simulator 3 (NS-3)," http://www.nsnam.org/.

[14] "DFN-Verein," http://www.dfn.de/.

[15] "DFN-Verein: DFN-NOC," http://www.dfn.de/dienstleistungen/dfninternet/noc/.

[16] A. Compagno, M. Conti, P. Gasti, and G. Tsudik, "Poseidon: Mitigating interest flooding DDoS attacks in named data networking," *arXiv preprint arXiv:1303.4823*, 2013.

[17] C. E. Leiserson, R. L. Rivest, C. Stein, and T. H. Cormen, *Introduction to algorithms*. The MIT press, 2001.

[18] M. D. Atkinson, J.-R. Sack, N. Santoro, and T. Strothotte, "Min-max heaps and generalized priority queues," *Communications of the ACM*, vol. 29, no. 10, pp. 996–1000, 1986.

[19] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in named data networking," in *Proceedings of the IFIP Networking Conference*, 2013.

[20] "LBNL's Network Research Group. arpwatch, the ethernet monitor program; for keeping track of Ethernet/IP address pairings." ftp://ftp.ee.lbl.gov/arpwatch.tar.gz.

[21] "Snort Project, The. Snort: The open source network intrusion detection system." http://www.snort.org.

[22] S. Y. Nam, D. Kim, and J. Kim, "Enhanced ARP: preventing ARP poisoning-based man-in-the-middle attacks," *Communications Letters, IEEE*, vol. 14, no. 2, pp. 187–189, 2010.

[23] Z. Trabelsi and W. El-Hajj, "Preventing ARP attacks using a fuzzy-based stateful ARP cache," in *Proceedings of the IEEE International Conference on Communications*. IEEE, 2007, pp. 1355–1360.

[24] A. Klein, "Bind 8 DNS cache poisoning," 2007.

[25] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "RFC 4033: DNS security introduction and requirements, march 2005."

[26] P. Mockapetris, "RFC 1035: Domain names: implementation and specification (november 1987)," 2004, http://www.ietf.org/rfc/rfc1035.txt.

[27] H.-M. Sun, W.-H. Chang, S.-Y. Chang, and Y.-H. Lin, "DepenDNS: Dependable mechanism against DNS cache poisoning," in *Cryptology and Network Security*. Springer, 2009, pp. 174–188.

[28] R. Bassil, R. Hobeica, W. Itani, C. Ghali, A. Kayssi, and A. Chehab, "Security analysis and solution for thwarting cache poisoning attacks in the domain name system," in *Proceedings of the 19th International Conference on Telecommunications*. IEEE, 2012, pp. 1–6.

[29] R. Perdisci, M. Antonakakis, X. Luo, and W. Lee, "WSEC DNS: Protecting recursive DNS resolvers from poisoning attacks," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks*. IEEE, 2009, pp. 3–12.