 CMC, vol.56, no.3, pp.483-499, 2018

Mitigating Content Caching Attack in NDN

Zhiqiang Ruan[1, *], Haibo Luo[1], Wenzhong Lin[1] and Jie Wang[2]

Abstract: Content caching is a core component in Named Data Networking (NDN), where content is cached in routers and served for future requests. However, the adversary can launch verification attack by placing poisoned data into the network with a legitimate name and allow the routers in the delivery path to frequently identify the content. Since NDN employs digital signature on each piece of content, verifying all content will exhaust routers' computational resources due to the massive data in the network. In this paper, we propose a selective verification scheme upon the contents that are hit in the content store and allow the contents that are not actually served to be unverified. We also consider the redundant verification of popular content and incorporated in our design to lessen the re-accessing overhead. Analysis and performance results show that the proposed scheme can greatly mitigate the risk of content verification attacks and save the computational resources of relay nodes.

Keywords: NDN, content cache, security, verification attack.

1 Introduction

Named Data Networking (NDN) is a network architecture for the future Internet, and can cooperate with the Internet of Things (IoT) to handle problems existing in the current TCP/IP network architecture [Zhang, Clafty, Crowley et al. (2014)]. In particular, content in NDN has hierarchical name, the request/interest packet is routed by the name and can be served either by the intermediate nodes or content source that have a copy of the content. The whole communication procedure is driven by the consumer and supported by a Content Store (CS), a Pending Interest Table (PIT), and a Forwarding Information Base (FIB) structure in nodes (i.e. NDN routers).

Since routers in NDN hold copies of the content that they relay, popular content can be distributed over the network and users can easily access to the content via content name. According to Cisco visual networking index in 2016 [VN Index (2016)], the global traffic will increase about threefold in the next five years, reaching 194 EB per month by 2020. Under such circumstances, in network caching and accessing based on named data can provide efficient content retrieval and enable NDN successfully minimize the amount of traffic over the link [Yuan, Crowley and Song (2017); Song, Yuan, Crowley et al. (2015)].

[1] Department of Computer Science, Minjiang University, Wenxian Road No.1, Fuzhou, Fujian, 350108, China.

[2] Department of Computer Science, University of Massachusetts Lowell, One University Avenue, Lowell, M. A., 01854, USA.

[*] Corresponding Author: Zhiqiang Ruan. Email: rzq_911@163.com.

Despite these great benefits, NDN has potential problem in terms of security. Specifically, in-network caching regards each node as a possible content provider, malicious users can inject poisoned content into the network by any legitimate name. Once the polluted content is saved on the CS of a node, it is possible to spread to the entire system. As a result, caches are polluted by bad data. To solve this problem, NDN appends a digital signature on the content and can be verified either by end-hosts or any routers. Users may reject the content because of signature mismatch, while the network is nearly unaware of the problem due to the computational burden of content verification. Thus, subsequent requests may continue to be answered by bogus content and constitute a denial of service (DoS) attack [Gasti, Tsudik, Uzun et al. (2013)]. Furthermore, routers cannot afford to verify all content, due to hundreds of Gbps content in the network and verifying of them will introduce huge overhead [Xie, Widjaja and Wang (2012)].

In this paper, we present a light-weight solution to detect verification attack at routers and identify the vulnerable request. The proposed solution exploits the relationship between the number of cache-hit events and the amount of serving content. If verification attack happens, network nodes may monitor more serving content than usual as much more unverified content is being accessed by the adversaries' requests. Thus, if the number of serving content increases above a certain threshold, the routers judge that they are under verification attack and start the attack alarms. Vulnerable requests are identified simply by counting the number of forwarded pieces of serving content per node. Our simulation study verifies that the proposed scheme can effectively detect verification attack and the malicious requests.

The contributions of this paper are summarized as follows.

1) Firstly, we propose the concept of selective verification at routers for stored/forwarded content and efficiently mitigate the risk of content poisoning attacks; the proposed scheme can work with most of the existing security mechanisms and align with the basic architecture of NDN.

2) Secondly, we present a simple mechanism for monitoring the amount of serving content and the cache-hit ratio, which helps to detect the verification attack and block the routers serving for the vulnerable fetches that directly connected to them. Furthermore, we treat the path of poisoned content as a bad forwarding choice and guide the selection of a better forwarding path using NDN's forwarding strategy.

3) Finally, we design an efficient algorithm for content poisoning attack and demonstrate that the proposed algorithm can mitigate such attack without degrading the QoS (i.e. latency). According to the simulation results, the verification overhead can be reduced over 30% and the verification efficiency can be improved by up to 50% under the cache replacement policy.

The reminder of this paper is arranged as follows. Section 2 surveys the state of the art in mitigation mechanisms on poisoned content. Section 3 gives the problem formulation. Section 4 illustrates the design detail of the proposed scheme followed by the analysis in Section 5. Section 6 shows the performance evaluation and Section 7 concludes the paper.

2 Related work

Recently, content poisoning mitigation mechanisms have become an active research topic. Dibenedetto and Papadopoulos [Dibenedetto and Papadopoulos (2016)] used the exclusion filter in interest packet to determine content that cannot satisfy the interest. However, there are several drawbacks remain to be solved. First, each consumer has to request/exclude separately to identify the desired data, as exclusion knowledge is limited to each consumer. Second, attackers can produce bogus data to an un-scalable exclude filter. Third, since exclusion is unable to declare any finer granularity, it cannot help consumers to retrieve data by a less name prefix.

Another poisoning mitigation approach is for interests to declare the content's expected publisher. NDN packet specification supports a *KeyLocator* field and the NDN name or digest (i.e. location) of its publisher's certificate is included in interest packet [Compagno, Conti, Ghali et al. (2015)]. Since a certificate is a sort of label that carries a set of keys, it could be requested in the same way when other content is retrieved. The legitimacy of the signer can be determined by encoding the identity of a key into a certificate name. Yu [Yu (2015)] constructed a hierarchical trust model, where a chain of keys or authorities form a hierarchy that is rooted at trust anchors, which are trusted by all verifiers. Hamdane et al. [Hamdane, Boussada, Elhdhili et al. (2015)] introduced hierarchical identity based cryptography for security and trust in named data networking. Research shows that developed a verifiable diversity ranking search method over encrypted outsourced data. However, neither of them is effective because an attacker can simply use the correct key information in malicious packets.

Kim et al. [Kim, Uzun and Wood (2015)] advocated authenticating data packets until they are served from a node's CS. However, the underlying forwarding problem is not addressed and would result in the poisoned content being re-requested. Ghali et al. [Ghali, Tsudik and Uzun (2014a, 2014b)] suggested that all interests specify the publisher via key digest and include the associated public key to each data packet. Later, routers verify each data packet with the attached key and ensure the key matches what is specified by the requesting Interest. However, since each router has to verify all data packet on the fly, which limits the application in reality.

Some researchers represent the content by a unique hash digest, and specify interests with digests that connected to the requested content by their exact names, such that the content poisoning attacks can be restricted in a small range. Gasti et al. [Gasti, Tsudik, Uzun et al. (2013)] linked every content to its predecessor by placing its digest in the predecessor's payload, which significantly reduced the verification overhead. However, it incurs other problems of trust management and overcoming inter-packet dependency. Research shows that an abnormal content feature sequence prediction approach for DDoS attacks in future generation internet, but it has the same problem of detecting all content sources.

On the contrary, Baugher et al. [Baugher, Davie, Narayanan et al. (2012)] and Kurihara et al. [Kurihara, Uzun and Wood (2015)] classified content into multiple catalogs and attempt to improve the verification efficiency in the whole process of content publication and retrieval. They define collections for content by their name and digest, and the publishers are set to sign a limited number of manifests rather than individual content. Consequently, the users can retrieve desired data with exactly name, avoiding content

poisoning attack as well as perform fewer signature verifications. Unfortunately, this approach is exclusively probabilistic and unable to adapt to bad content.

Ghali et al. [Ghali, Tsudik and Uzun (2014a, 2014b)] proposed the concept of self-certifying name and design a method that exploits users' feedback to eject polluted content. However, it is risky because valid content are also excluded from the CS upon the fabricated feedback. Bianchi et al. [Bianchi, Detti, Caponi et al. (2013)] presented Lossy Caching strategy, where content is verified and cached based on a certain probability. Routers minimize verification overhead by lowering the probability, while the probability also influences the cache hit ratio and the freshness of network nodes. When the probability is low, verification and caching are favor to more popular content, while caching is probably fill with outdated content. Thus, it is difficult to find the optimal probability value. Moreover, Lossy Caching is strongly coupled with probabilistic caching, and it is prohibit from using this scheme to different types of cache repracement policies.

3 Problem statement

In verification attacks, forged data packet is placed in the CS of routers. To match with the interest, the faked content has a valid name, but its signature or payload is fabricated. When requests for the content pass by the target router, they are served by polluted content in the CS rather than forwarded to the content source, when the poisoned content relayed back to the user, it locates in the CS of all involved routers. As a result, the CS is filled with useless content. What is more, subsequent requests for desired content cannot success in a single attempt before cross over the attacked router. After recognizing the disproof of the received content, a user resubmits the request with the hash value of the polluted content in the exclude field, which incurs extra delays and verifying overhead at routers.

The implement of content poisoning attack is manifold. A typical example is that the attackers use two end-hosts, one is the client node that issues an interest and the other is the server node which provides the poisoned content. When the client requests content, a unique interest is generated and forwarded to the valid content source by the routers. From then on, the server injects the poisoned content into the network at arbitrary or illogical step. It is not mandatory for routers to check the arriving request; however, the adversary may only attempt to consume the PIT entry by inserting poisoned content into the CS and delivered back to the user. Even if the valid content reaches to the router afterward, it is immediately discarded as no pending interest is matched in the PIT.

Another case for content poisoning attack is the routers may copy interests and assign them via several requests to search the nearest duplicate of the content [Chiocchetti, Perino, Carofiglio et al. (2013)]. In this case, interests can be transmitted via randomly selected requests before arriving at the attacker's server. So, poisoned content leakages into the network and pollutes the CS of middle routers. The first guard to these types of attacks may be secure routing. As for more in-depth measures, one need to provide effective and security mechanisms that take the benefit of in-network caching of NDN into account.

Note that content poisoning is substantially different from cache poisoning; in particular, content poisoning is still exist no matter whether there were in-network caches, as the adversary can still serve bad data. While cache pollution aims to deplete local cache

space, the attackers can intentionally request the unpopular content and fill them into the cache. As a result, popular content is ejected from the cache and the benefit of in-network caching is discounted. Since cache pollution attack use valid content, there is no need to check the legality of the content itself.

4 The proposed scheme

We carry out the concept of examine on cache-hit, that is, all content arriving at the router are placed in the CS without signature verification. Only when a cache-hit appears in the CS, then the serving content is authenticated before flowing into the network. Compared to complete verification scheme (referred to as basic scheme), our approach saves a large amount of computational resources for by-passing content, and it nicely avoids the malicious effects of poisoned content. In other words, poisoned content in the CS is either simply removed from the CS without any negative impacts or authenticated before affecting the network. Hence, there is make no sense to intervene the CS by multiple pending interests for poisoned content spread out over the network, because data packet that has multiple pending interests are considered as serving content and is check before being forwarded. Nevertheless, this approach would increase the access delay of popular data. One alternative is to allow the poisoned content to spread over the network, although this enables the poisoned content endanger the network caches, it still can be detected immediately and wiped off by new requests from users.

In general, popular content is frequently accessed in the CS. To avoid repeated authentication on the following cache-hits, a label or flag is set in the CS to indicate that the content is already passed the verification and this flag stays in the CS with the content. It is possible that the remaining cache space at router is not enough to hold popular content, in this case, popular content may be ejected from the CS, while this content is re-inserted into the CS on the next data retrieval, the content has to be check repeatedly when a cache-hit occurs after each insertion. This shortcoming is also mentioned in the original NDN security architecture. To solve this problem, we employ the technique of Segmented Least Recently Used (SLRU) [Karedla, Love and Wherry (1994)] to the CS.

SLRU divides the cache into a protected section and an unprotected section, each segment applies LRU policy individually. If a component on the unprotected section is visited, it is moved to the protected section and can stay longer than those objects in the unprotected section. Given preference to repeatedly access content, SLRU can greatly improve the cache-hit ratio. If we apply SLRU in our algorithm, verified content is migrated to the protected section and cannot be excluded by going through content. Even if the content moves out of the protected section, according to LRU cache replacement strategy, it is first arranged into the unprotected section and then removed from the cache. Therefore, authenticated content always has a higher chance of being re-visited, which significantly reduces the overhead of repeated verifying popular content.

We use the metric λ to analyze the efficiency of our scheme, and it is defined as

$$\lambda = M_k / M_v \tag{1}$$

where M_k is the number of verification carried out for poisoned content, M_v is the number of verification implemented for all content, and k is the ratio of requests for poisons content out of all requests. M_k is equal to $dk\Delta t$. Since we only perform validation for unauthenticated serving content, if the hit rate for unverified content in the CS is h_l, then in the time interval Δt, M_v can be denoted as $dk\Delta t + h_l(1-k)\Delta t$, where d is the arriving rate of request. Note that a cache-hit happens upon poisoned content is reported by the re-request or the popular content is re-visited. Hence, the value of λ in this work, λ_l, is

$$\lambda_l = \frac{dk}{dk + dh_l(1-k)} = \frac{k}{k + h_l(1-k)} \tag{2}$$

To calculate h_l, we assume that the popularity of total content (M pieces) follows the Zipf-Mandelbrot distribution:

$$p_M(i) = \frac{\mathrm{B}}{(i+m)^z} \tag{3}$$

where $\mathrm{B} = (\sum_{i=1}^{M} \frac{1}{(i+m)^z})^{-1}$, h_l is the probability that the content in the CS that has missed cache-hit prepares a cache-hit for the next request. Therefore, h_l is written by

$$h_l = \sum_{\forall i} p_M(i) p_H(i) p_L(i) \tag{4}$$

where $p_H(i)$ and $p_L(i)$ are the probabilities of a cache-hit and cache-miss for the content i, respectively. $p_H(i)$ can be further written by

$$p_H(i) = 1 - k^{-p_M(i)S} \tag{5}$$

where S is the capacity size of CS and it can be calculated by

$$S = \sum_{\forall i} (1 - k^{p_M(i)t}) \tag{6}$$

If SLRU is used for our scheme, given the new arrival request, h_l is corresponds to the conditional probability that content is moved from the unprotected section to the protected section in the CS. In short, h_l is equal to the hit rate in the unprotected section. Suppose the ratio of the protected section in the CS is β, then contents with the rank i ($\leq \beta S$) are arranged in the steady state from the protected section. Thus, h_l is calculated by Eq. (5) and Eq. (6) under the normalized distribution of content popularity for $M - \beta S$ pieces and a CS with a size of $(1-\beta)S$.

In order to solve verification attack, the first step is to estimate whether the routers are under verification attack or not. One of the most prominent feature of verification attack is a large number of unverified data packet is maliciously fill into the CS to produce cache-hit events. In view of this, verification attack might be detected by utilizing the correlation between the number of cache-hit events and the amount of verifications that have been performed.

Given the distribution function of the content popularity, the number of cache-hit events can be represented by *hdt*, where *t* is the monitoring interval and *d* is the request arriving rate. *h* is the cache-hit rate of all content, which is calculated by

$$h = \sum_{\forall i} p_M(i)\, p_H(i) \tag{7}$$

Algorithm 1: Detection of Pollution Attack

Input: Number of requests *r*, number of cache-hits *h*; number of verifications *v*, time interval Δt, threshold ε

Output: 1 (Success) or 0 (False)

```
1  begin
2    repeat
3      r=r+1
4      if cache-hit occurs then
5        h=h+1;
6        if first cache-hit for unverified content then
7        v=v+1;
8        endif
9      endif
10       calculate v/h
11     if v/h < ε then
12       update ε by (8)
13     else
14       go to the identification stage
15     endif
16     reset r, h, v
17   until Δt ≠ 0
18 end
```


The anticipative number of validations is *Hdlt*, where h_l is derived from Eq. (4). Accordingly, the ratio of cache-hit events out of all verifications can be denoted by the value of h_l / h, which should be stable no matter how the popularity rank of each content changes. However, if cache-hit events are manipulated by the attackers, the ratio of validations to cache-hit events grows abnormally and the observed content popularity distribution is untruthfulness. If the ratio reaches a certain threshold, routers know that they are under verification attack.

Obviously, the content popularity distribution function is not always constant, it more or

less skew by a small degree, in this case, a static threshold value is not work as a false positive error arises in the detection of verification attack. However, consider that verification attack changes the value of h_l / h much more radically than the popularity distribution function, thus, false negative errors will appear with a very low probability. To avoid false positive errors, the threshold value, ε, is set as

$$\varepsilon = \min(1, \theta \frac{F(h_l)}{F(h)}) \tag{8}$$

where θ is a constant larger than 1 and F is an exponentially-weighted moving average function. The procedure of detection verification attack is illustrated in **Algorithm 1**.

Algorithm 2: Identification Stage

Input: r, k $(r \le k)$

Output: 1 (Success) or 0 (False)

```
1 begin
2    typedef struct {int count; char* content;}
object
3  foreach i in r do
4     create object O
5     O.count=0
6     O.content=content
7     O is put into the CS
8  endfor
9  if cache-hit occurs then
10     O.count=O.count+1
11     r=lookup the PIT
12     i = index of r
13      if O.count==1 then
14          r[i]=r[i]-1
15      endif
16  endif
17 end
```


If the routers detect verification attack, they should disable the attack immediately. For our scheme, verification attack is first discovered at the node that is directly connected to the attacking router. After sensing the attack, the router switches to the identification mode to judge the vulnerable fetches (since the normal data requests are issued from the consumer side, we hereafter refer to these interests as fetches to distinguish them from reports). In the identification stage, the target router (victim) records the number of verified content to be forwarded per fetch. As mentioned earlier, to avoid redundant verification, a symbol or flag can be used to mark verified content. Therefore, routers

compute the amount of content that has been marked. If a specific request for verified content is forwarded excessive a given threshold, it is considered to be a vulnerable fetch. The detailed process is outlined in Algorithm 2.

The router can simply block the vulnerable fetches if the attacking nodes are end-hosts. If the attack is initiated from a captured router, blocking legitimate requests may cause users served by other content sources that are detoured farther away. From this point of view, unpopular yet legitimate content is filled up the CS, and verification attack has no different from cache pollution attack. The difference is cache pollution attack only issues a single request to ruin cache locality, while content poisoned attack must send out at least two same requests to impose verification overhead. Hence, verification attack can be effectively mitigated if the feasible solutions for cache pollution are used to the compromised router after the identification stage.

5 Analysis and discussion

5.1 Access delay

It is worth noting that the proposed scheme only performs one-time verification on the content upon cache hit in the CS, the subsequent accesses to the content do not need to check any more. To estimate the delay of the first access and the benefit of one-time verification, we use the access delay D and delay gain G to denote the retrieving time and the amount of time saved during content request and report.

Let ω_i and ω_i' be round-trip times of content i from a customer to the source and from the customer to a router, respectively. There are two cases: 1) If the content is not stored at the router, the average access delay, D, can be calculated by

$$D = \frac{\sum_{\forall i} p_M(j) d w_i}{d} = \sum_{\forall i} p_M(i) w_i \tag{9}$$

Otherwise, 2) If the content is stored at the router, the average access latency, D', is

$$\begin{aligned} D' &= \sum_{\forall i} p_M(i)(w_i p_L(i) + \omega_i' p_H(i)) \\ &= D - G_q \end{aligned} \tag{10}$$

where G_q is the delay gain, which is calculated by

$$G_q = \sum_{\forall i} p_M(i) p_H(i)(\omega_i - \omega_i') \tag{11}$$

Here, $(\omega_i - \omega_i')$ is a constant value, i.e. $\alpha\omega$, then

$$G_q = \alpha\omega \sum_{\forall i} p_M(i) p_H(i) = \alpha\omega\psi \tag{12}$$

where $\psi = \sum_{\forall i} p_M(i) p_H(i)$

If content is first hit and verified in the CS, the additional access delay, V_q, is given by

$$V_q = \frac{\delta h_l d}{d} = \delta h_l \tag{13}$$

where δ is the authentication delay. Clearly, $h_l \le \delta$, as $p_L(i) < 1$ for all i in Eq. (4). According to the measurement results in Gasti et al. [Gasti, Tsudik, Uzun et al. (2013)], a

signature verification using 1024-bits RSA on 1500 byte piece of content by Intel dual Core 2.6 GHz CPU takes about 80 μs. Since $\delta h_l << \alpha\omega\psi$, it has $V_q << G_q$. For a particular user, the access delay is only increased by δ, while it is much less than $\omega_i - \omega_i'$ and usually on the order of millisecond, while δ is on the order of microsecond. Therefore, the access delay is negligible under the proposed solution.

5.2 Asynchronous verification

A salient feature of the proposed scheme is that signature verification is decoupled with the caching operation, which means that routers can check the cached content anytime as long as they have extra computational capabilities. While the traditional NDN schemes are synchronous with the content caching process, that is, once the content is decided to cache on the router, it must be verified. Thus, our method can minimize the authentication time and reduce the unnecessary computation resources of the routers.

In fact, the proposed scheme is also favors to popular content. Assume that the content is managed in the form of queue in a router, for unverified content, the longer access interval between two requests indicates the less popular of the content. In other words, the access interval of popular content is much shorter than unpopular content. As a result, popular content can be arranged on the top of the queue. Thus, the proposed scheme improves the efficiency of verification.

6 Simulations

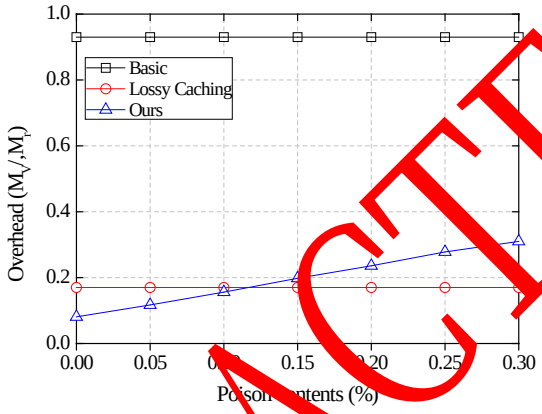
We evaluate the performance of the proposed scheme by using the ns-3 ndnSIM simulator [Mastorakis, Afanasyev and Zhang (2017)]. In the simulation, 100000 pieces of content are served by the server node, and their popularity follows the Zipf distribution with the parameter value, *z*. We assume that 1000 clients request content at a rate of one piece per second. The link bandwidth is set such that it is large enough to exclude the congestion effect. The CS size is varied from 500 to 5000.

Unless otherwise declared, all schemes are implemented using the naive LRU cache. Poisoned content at the server are simply generated with a given error probability. The clients immediately re-request the content (by generating a new interest) when they receive poisoned content. Simulation is performed for 24 h with each interval lasts for 10 min. We compare the proposed scheme with basic approach and Lossy caching [Bianchi, Detti, Caponi et al. (2013)]. In basic scheme, all unverified content are check before being inserted into the CS.

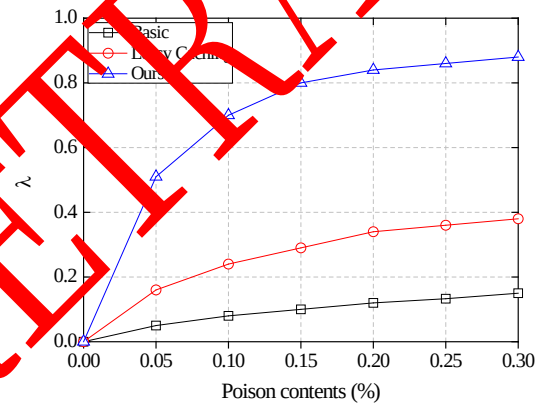
6.1 Verification cost

Fig. 1 investigates the impact of poisoned content on the performance of all methods, where we set *z*=0.8 in the Zipf distribution and fix CS size with 500, we record the values of verification overhead and λ by varying the percentage of poisoned content in the network. The verification overhead is calculated by M_v / M_r, where M_r is the number of arriving requests. The caching probability (*l*) in Lossy Caching is set to 0.1.

As we can see, even with different amounts of poisoned content, the verification overhead does not change in either the basic scheme or Lossy Caching (Fig. 1(a)). This is caused by the fact that verification is performed regardless of the content's state. The largest overhead is shown in the basic scheme and the verification overhead in Lossy Caching is determined by the caching probability. In the proposed scheme, however, the verification overhead increases in proportion to the amount of poisoned content. This is because more cache-hits occur in the CS due to re-request messages from clients that received poison content. Here, we emphasize that despite the increased overhead, the proposed scheme maintains a high level of λ, indicating that unnecessary verifications are effectively minimized, as shown in Fig. 1(b).

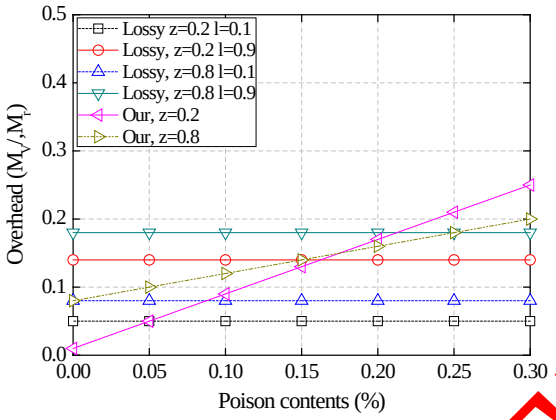


(a) Verification overhead

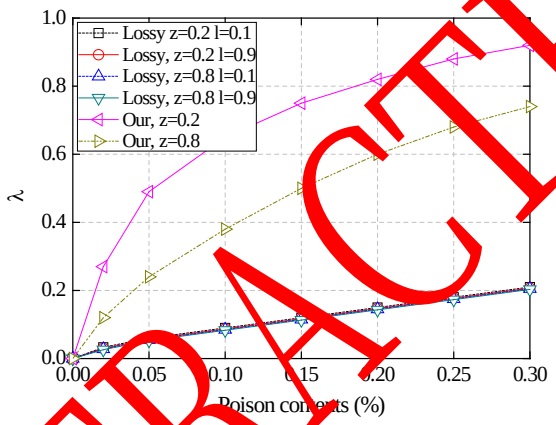


(b) λ

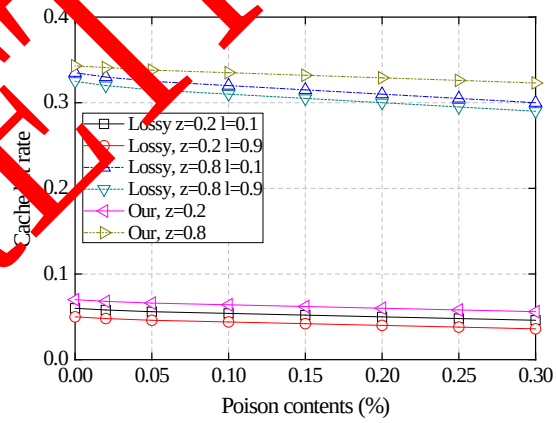
Figure 1: Performance of poisoned content



(a) Verification overhead



(b) λ



(c) Cache hit rate

Figure 2: Comparisons under dynamic content popularity

Fig. 2 compares the verification overhead, cache hit rate, and λ value with dynamic content popularity under our scheme and Lossy Caching. We adjust the ranks of content

popularity every 5 min via a random number ς. For content *i*, the rank is $i+\varsigma$ in the next 5 min. If $i+\varsigma > M$, modular arithmetic is used, i.e. $(i+\varsigma) \bmod M$. In the figure, *z*=0.2/0.8 is the content popularity in Zipf distribution, and *l*=0.1/0.9 is caching probability in Lossy Caching. As wee can see that the overhead of our scheme slightly grows as the increases of poisoned content, while the overhead of Lossy Caching is dependent on the value of *l*, which is similar to previous results in Fig. 1. However, our scheme has a larger value of λ, indicating that we can detect more poisoned content than alternatives. Note that the caching probability *l* is irrespective of the value of λ as content state does not rely on probabilistic caching. For the cache hit rate, when we fix *z*, a larger *l* results in a lower hit rate, and our scheme has a higher hit rate than Lossy Caching, this is because, with the increasing deviation of content popularity, the advantage of probabilistic caching is gradually disappeared.

6.2 Access delay

Fig. 3 investigates the average access delay of the users under poisoned content by different schemes. As we can see that the access delay of all methods, except for the basic scheme, are grow in proportion to the ratio of poisoned content, and our scheme incurs lowest delay for the user, which can reduce at least 30% and 50% of latency compared to the basic method and Lossy Caching, respectively. The reason for the basic scheme is straight, as all the cached content should be verified before transmitting to the next hop regardless of the content's state on the routers. For Lossy Caching, the conclusion is similar to the previous simulations, the verification time is depends on the value of *l*, while the access time in the proposed scheme increases slightly with the amount of poisoned content.

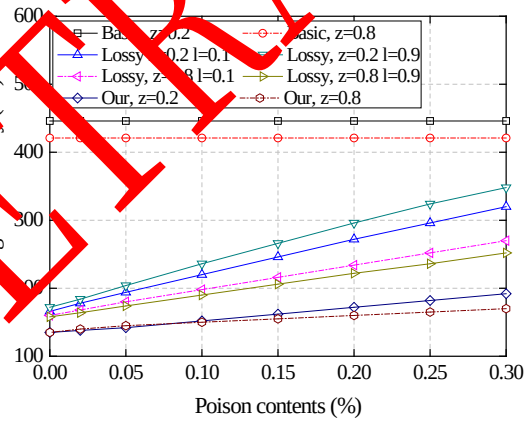
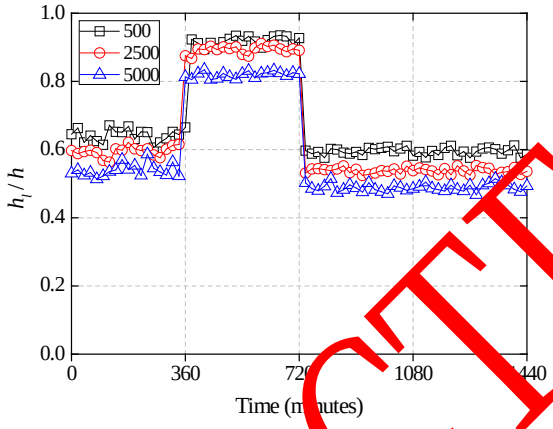


Figure 3: Average access delay under poisoned content

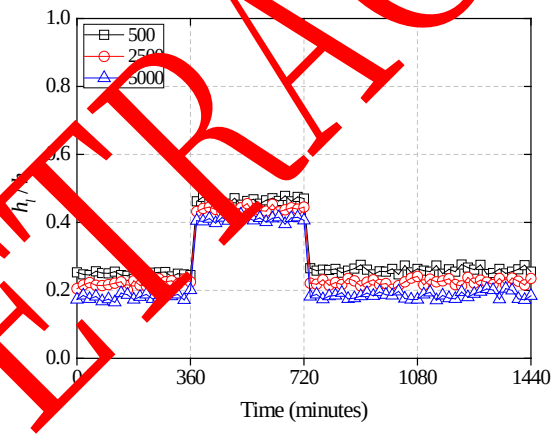
6.3 Impact of system parameters

Fig. 4 plots the values of h_l / h every 20 min with different sizes of the CS. As can be observed that without verification attack, the values of h_l / h is about 0.6 (Fig. 4(a)) and 0.2 (Fig. 4(b)). It is noted that the values in Fig. 4(b) are smaller than those in Fig. 4(a)

because popular content is more frequently accessed when z=1.0. After 6 h, the value of h_l / h becomes distinctively larger for a period of 360 min. As previously explained, the traffic that is manipulated by the attacker changes the original popularity distribution of the content; this is successfully sensed by the value of h_l / h. If the value of h_l / h increases above the threshold, as presented in Eq. (8) where the value of θ is set as 1.5, the routers move into the identification stage to find the vulnerable fetches.



(a) z=0.7



(b) z=1.0

Figure 4: The ratio of verification under pollution attack

Fig. 5 shows how much serving content is delivered via each request every 20 min. Before verification attack, an average of 8~12 pieces and 10~15 pieces of content are forwarded per request, when z=0.7 and z=1.0, respectively. After the attack is launched, however, 120 pieces and 150 pieces of content are served via vulnerable fetches, while the other requests still forward a similar amount of serving content. Therefore, the attacker is effectively blocked when the vulnerable fetches are disabled. It is noted that operations in the identifying stage are triggered after verification attack is sensed in order to minimize the overhead.

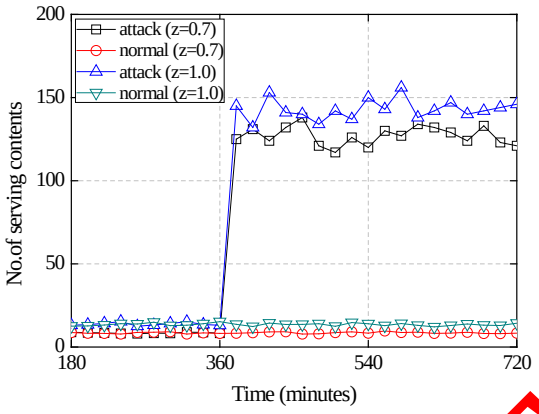
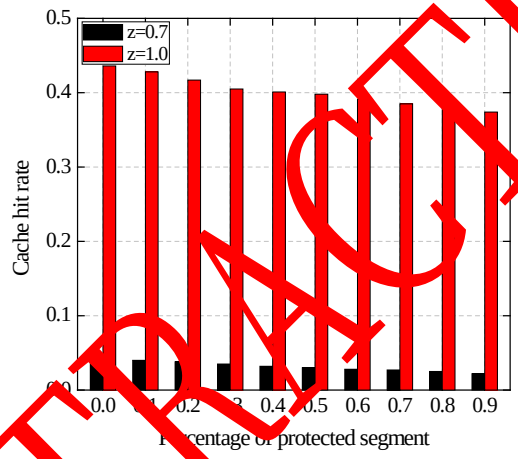
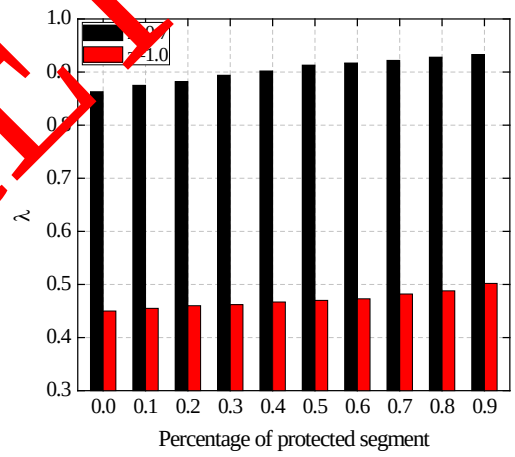


Figure 5: Identifying of malicious requests



(a) Cache hit rate



(b) λ

Figure 6: Impact of SLRU on the proposed scheme

Fig. 6 examines the effect of SLRU in three schemes. We set the proportion of poisoned content to 0.1. The overall size of the CS is fixed as 5000, and the proportion of the protected segment in the CS is varied from 0 to 0.9. As we can see, with the size of the protected segment increases, the time during which content stay in the unprotected segment before eviction becomes shorter. As a result, the cache hit rate decreases. However, the verification efficiency, λ, grows in proportion to the size of the protected segment, indicating that verified content is re-used more frequently.

7 Conclusions

This paper addresses the problem of content pollution attack in NDN and proposes a light-weigh solution to mitigate it. We propose the concept of selective verification on the cached content, and avoid meaningless verification for by-passing content and by favoring already verified content, which saves a large amount of computational resources. Performance results show that malicious effects from poisoned content in the CS are perfectly prevented. In the future, we will investigate the application of verification attack in a real system. In future works, we will explore the implementation of the proposed scheme in coupling with the interest/data forwarding system.

Acknowledgement: This work was supported by the Natural Science Foundation of Fujian Province (Grant number: 2018J01544); the Key Project of Natural Foundation for Young in Colleges of Fujian Province (Grant number: JZ160466); the Scientific Research Program of Outstanding Young Talents in Universities of Fujian Province; the Scientific Research Project from Minjiang University (Grant numbers: MYK16001 and MYK17025), and Fujian provincial leading project (Grant number: 2017H0029).

References

Baugher, M.; Davie, B.; Narayanan, A.; Oran, D. (2012): Self-verifying names for read-only named data. *IEEE International Conference on Computer Communication Workshops*, pp. 121-126.

Bianchi, G.; Detti, A.; Caponi, A.; Blefari-Melazzi, N. (2013): Check before storing: what is the performance price of content integrity verification in LRU caching? *ACM Sigcomm Computer Communication Review*, vol. 43, no. 3, pp. 59-67.

Chiocchetti, R.; Perino, D.; Carofiglio, G.; Rossi, D.; Rossini, G. (2013): Inform: A dynamic interest forwarding mechanism for information centric networking. *ACM Sigcomm Workshop on Information Centric Networking*, pp. 9-14.

Compagno, A.; Conti, M.; Ghali, C.; Tsudik, G. (2015): To nack or not to nack? Negative acknowledgments in information-centric networking. *IEEE International Conference on Computer Communication and Networks*, pp. 1-10.

Dibenedetto, S.; Papadopoulos, C. (2016): Mitigating poisoned content with forwarding strategy. *IEEE International Conference on Named-Oriented Mobility*, pp. 701-706.

Gasti, P.; Tsudik, G.; Uzun, E.; Zhang, L. (2013): Dos and ddos in named data networking. *ACM Sigcomm Computer Communication Review*, vol. 44, no. 3, pp. 66-73.

Ghali, C.; Tsudik, G.; Uzun, E. (2014a): Network-layer trust in named-data networking. *ACM Sigcomm Computer Communication Review*, vol. 44, no.5, pp. 12-19.

Ghali, C.; Tsudik, G.; Uzun, E. (2014b): Needle in a haystack: mitigating content poisoning in named-data networking. *NDSS Workshop on Security of Emerging Networking Technologies*, vol. 34, no. 1, pp. 68-73.

Hamdane, B.; Boussada, R.; Elhdhili, M. E.; Fatmi, S. (2017): Hierarchical identity based cryptography for security and trust in named data networking. *IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 226-231.

Karedla, R.; Love, J. S.; Wherry, B. G. (1994): Caching strategies to improve disk system performance. *Computer*, vol. 27, no. 3, pp. 38-46.

Kim, D.; Nam, S.; Bi, J.; Yeom, I. (2015): Efficient content verification in named data networking. *ACM International Conference on Information-Centric Networking*, pp. 109-116.

Kurihara, J.; Uzun, E.; Wood, C. (2015): An encryption based access control framework for content-centric networking. *IFIP Networking Conference*, pp. 1-9.

Mastorakis, S.; Afanasyev, A.; Zhang, L. (2017): On the evolution of ndnSIM: An open-source simulator for NDN experimentation. *ACM Sigcomm Computer Communication Review*, vol. 47, no. 3, pp. 19-33.

Song, T.; Yuan, H.; Crowley, P.; Zhang, B. (2015): Scalable name-based packet forwarding: From millions to billions. *ACM Conference on Information-Centric Networking*, pp. 19-28.

VN Index (2016): Cisco visual networking index: Forecast and methodology, 2015-2020. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking index-vni/complete-white-paper-c11-481360.html.

Xie, M.; Widjaja, I.; Wang, H. (2012): Enhancing cache robustness for content-centric networking. *IEEE International Conference on Computer Communications*, pp. 2426-2434.

Yu, Y. (2015): Public key management in named data networking. *Technical Report, NDN-0029*. University of California, Los Angeles, CA, USA.

Yuan, H.; Crowley, P.; Song, T. (2017): Enhancing scalable name-based forwarding. *ACM/IEEE Symposium on Architectures for Networking & Communications Systems*, pp. 60-69.

Zhang, L.; Claffy, K.; Crowley, P.; Papadopoulos, C.; Wang, L. et al. (2014): Named data networking. *ACM Sigcomm Computer Communication Review*, vol. 44, no. 3, pp. 66-73.