ATLANTIS PRESS International Journal of Networked and Distributed Computing, Vol. 6, No. 3 (July 2018) 174 - 184

A Triggered Delay-based Approach against Cache Privacy Attack in NDN

Naveen Kumar, Ashutosh Kumar Singh, Shashank Srivastava

Department of Computer Science & Engineering, Motilal Nehru National Institute of Technology Allahabad, Allahabad, Uttar Pradesh 211004, India.

E-mail: nk10121989@gmail.com, ashuit89@gmail.com, shashank12@mnnit.ac.in

Abstract

Content caching is one of the most significant features of Named Data Networking (NDN) that improves the performance. However, this feature makes the cache vulnerable to attacks that determine the recent cache access pattern. In cache privacy attack, an attacker can probe request and determine if the received content is cached or not, by simply observing the time difference between the requested and the received data. Existing solutions apply delay whenever the data is accessed from the cache. These approaches mitigate attack to some extent but compromise the performance of NDN. To overcome this issue, a counter scheme has been proposed in this article that detects the attack pattern at the gateway router itself and triggers the countermeasure in case of attack. The triggered-based approach delays the data accessed from the cache, only when the attack is detected instead of each time when the data is accessed from the cache. The proposed approach has been compared with an approach that induces a random delay in case of the cache hit. The results prove that the triggered delay-based approach is better than the random delay approach in terms of average delay.

Keywords: Cache Privacy Attack, CPA, Named Data Networking, NDN

1. Introduction

According to forecast conducted by Visual Networking Index 2016, global Internet Protocol (IP) traffic per month will reach 194 Exabytes and IP video traffic will be 82% of all traffic by the year 2020[6]. TCP/IP was developed to overcome the problem of communication between users. In current scenario, Internet is mainly used to fetch data. Users do not bother about who is satisfying the request until the data is fetched from the authenticated publisher. Information Centric Networking (ICN)[5] was introduced to handle requirement of the current network. Some popular ICN-based networks are data-oriented network architecture[8], content mediator architecture for content-aware networks[14], Content Centric Networking (CCN)[7], and Named Data Networking (NDN)[15].

NDN is one of the most promising ICN candidates among all the ICN type networks. NDN has content-centric network architecture rather than a host-centric. It uses the name of the data for forwarding, routing, and fetching content. The consumer can request a content by using interest packet which can be satisfied by a data source (publisher or router having that data) using the data packet. Additionally, the data is cached by each in-between routers.

Content caching is the fundamental feature of NDN that reduces network congestion, optimizes bandwidth utilization and provides fast access to the content. However, this feature enables attackers to access the pattern of the cache miss or hit. In Cache Privacy Attack (CPA), an attacker tries to

find out recently accessed private content from the gateway router to which it is directly attached. The attacker compiles a list of privacy-sensitive unpopular contents. The attack timing and the list is intelligently decided so that the attacker can associate these content to user(s). After compiling the list of the contents, the attacker calculates the hit time of the cache for the access router by requesting same content twice. The first time the content is accessed from the data source (publisher or the router having that data packet) and the second time the content is accessed from the cache. The time taken when the data is accessed second time is the hit time for the cache. After knowing the hit time, the attacker requests each content in the list one by one to find out whether it is accessed from the cache or not by comparing the delay with the hit time. After knowing whether the content is hit or not the attacker can find out the type of content accessed by the users and the timing of accessing those contents.

Solutions proposed by some of the authors[14,7,15,16] have degraded this attack to some extent, but with the degradation of the NDN performance. A counter technique for the mitigation of CPA is proposed which utilizes the attack pattern for the detection of the attack. After the detection, a counter is triggered to inform the router to apply delay only to the malicious namespace. Thus our approach only targets the malicious namespace rather than all the contents. The main contributions of this paper are

- Implemention of CPA using ns3[13] based ndnSIM[3] simulator on the linear topology.
- Development of a detection approach that counter CPA proposed by Lauinger et al.[9].
- A delay based counter measure which trigger after the detection of CPA.
- Comparison of the proposed approach with the approach in which random delay is applied every time on cache hit.

The rest of the paper is organized as follows. Section 2 describes NDN architecture in brief. Section 3 presents related work on CPA. Section 4 describes CPA in NDN. Section 5 shows the attack modeling, attack detection and countermeasure. Section 6 illustrates the result of attack detection. Finally, section 7 concludes this article along with scope of future work.

2. NDN Architecture

In NDN, the name of the data plays a significant role as it helps in forwarding, and routing of the content. NDN uses human-readable hierarchical names for naming content. Each name has variable length string components which are separated by "/" as the delimiter. For example */ucla/new/video3.mp4*. There are two types of packets in NDN, i.e., interest packet and data packet these packets are used for the communication. Interest packet is used to send the request, and the data packet is the reply for that interest packet. In NDN data source can be a publisher (or the router having data packet in its Content Store). Each data packet is signed by the publisher which can be verified by the consumer using publisher's public key given by the key locator field of the data packet.

Each router maintains three kinds of data structures, *i.e.*, Pending Interest Table (PIT), Forwarding Information Base (FIB), and Content Store (CS). PIT stores all the metadata of all the incoming unsatisfied interest packets. Each PIT has TTL field which specifies the time after which the PIT entry becomes invalid.

If data packet corresponding to the entry is not received before expiration of TTL, then the PIT entry is flushed. The router checks FIB to find out where to forward the interest packet when the data packet is not found in the CS. Whenever a new data come from the data source, it gets cached in CS according to the replacement strategy like Least Recently Used (LRU), or Least Frequently Used (LFU).

The consumer specifies its request by forming an interest packet containing the name of the data. On receiving the interest packet, the router first checks its CS. If the data packet corresponding to the interest packet is present in CS then it is forwarded back to the consumer, else the name in the interest packet is checked in the PIT. If a matching entry is found in the PIT then the incoming interface of the interest

packet is appended to the PIT entry, else a lookup is performed in the FIB table. The FIB decides the outgoing interface on the basis of the longest prefix match, and a new entry for the interest packet is created in the PIT. If there is no match found in FIB, then the router sends either drop the interest packet or send a negative acknowledgment back to the sender depending upon the router's policy.

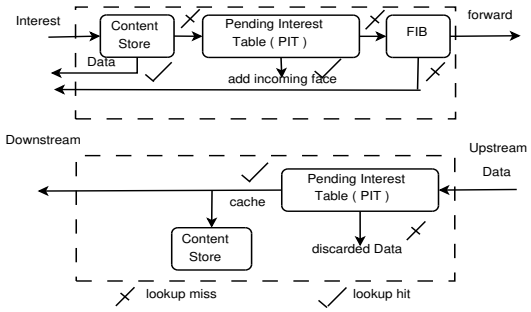

Fig. 1. Forwarding pipeline in NDN

On receiving a data packet, the router first checks if its entry present in the PIT or not. If the entry is found then the router forwards the data packet through all the interfaces specified by the incoming interface field of matching PIT entry. Additionally, the data packet is cached in CS using caching strategies like LRU or LFU. If no entry found corresponding to the data packet, then the packet is dropped. The forwarding pipeline is given in Fig. 1

3. Related Work

Lauinger et al.[9] proposed request monitoring attack in which an attacker performs an attack by probing interest packet at frequency f_p called probing frequency. This f_p should be greater than or equal to $\frac{1}{tc}$ for 100% detection rate, here t_c is the average period for which a data packet remains in CS. Several countermeasures like tunneling, disabling the scope field, etc., have been proposed these approaches reduce CPA to some extent but they compromise NDN performance too.

Mohaisen et al.[10,11] have given three different approaches based on applying delay for time t in the case of a cache hit. These are 1) vanilla approach, 2) efficient approach, and 3) low granularity approach. In the first approach, the edge router maintains the record of the content's name and the number of times the content requested by a consumer. If a consumer requests a data that the consumer requested previously then no delay is applied. Otherwise, a delay is applied based on the RTT of that content. In the second approach, the router maintains the record of the content's name and the number of times the content requested from an interface of the router. This approach incurs less storage overhead for the router. The third approach takes the advantages of both the above approaches by maintaining both the records, i.e., the number of times the content requested from an interface and the user. These three approaches were proposed for ICN that cannot be applied to the NDN as in NDN there is no User-ID and no concept of a layer-2 device.

Acs et al.[1] proposed three delay based approaches which apply a delay on replying data packets for the first k requests of privacy-sensitive contents. These approaches are 1) Non Private Naive, 2) Uniform Random Cache, and 3) Exponential Random Cache. In the first approach, this k is fixed for all the content in the cache. In the second approach, this k is chosen using a uniform random variable which gives a value between $(0,k]$. In the third approach, this k is chosen using an exponential random variable. They have extended their work in which they have given a proof of their approach[11]. This approach is based on maintaining trade-off between privacy and caching. In this approach, each router has to maintain the state of all the contents that are present in CS. Therefore, it gives extra processing and storage overhead to the router.

Zhang et al. [17] given a security mechanism based on adding a dynamic password. This approach allows a legal consumer to add some random number as nonce calculated by an algorithm known to the consumer. Receiving router may accept or reject this request based on the added number. Under this scheme, the router has to perform extra computation like checking request field for each request which may lead to additional overhead. Ntuli and Han[12] proposed a scheme to detect cache snooping in NDN on the basis of the high-interest rate in short time. Dogruluk et al.[4] proposed a countermeasure based on the above detection in which the router applies delay fixed, random, or cryptographic delay.

Most of the previous approaches try to avoid CPA by adding additional delay. However, there is very less work done on the actual implementation of attack and its countermeasure. Our approach tries to counter the CPA by detecting the attack based on the attackers' pattern of probes and then applying countermeasure.

4. Cache Privacy Attack

The main issue regarding cache privacy is deciding whether the content is private or public. Privacy of content depends on whether it can be linked to a particular context (such as political view, religious beliefs, etc.), user, the location at a given time. The entity (either user or publisher) which will decide the privacy of a data and how this privacy will be implemented (how private and public content is differentiated so that the router reacts according to it) is also a crucial issue in NDN.

Besides these issues, if the content is private, then it must be handled in such a way that attackers could not detect whether the content is recently requested or not. Cache privacy can be maintained by disabling router's cache, but that would result in a decrement of NDN performance. A good solution for this attack is to make difficult for an attacker to link privacy sensitive content to a user(s).

There are mainly two types of CPA *i.e.*, timing-based attack and object discovery attack. Our work is mainly focused on timing based attack.

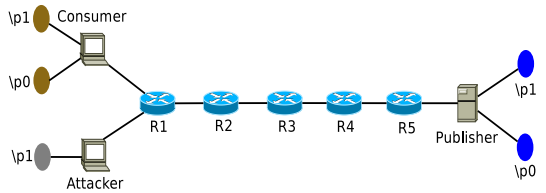

Fig. 2. Detection Model

4.1. Timing Based Attack

In this attack, an attacker tries to find out the recent access to a privacy-sensitive content from the nearby cache. The attack is performed by compiling a list of privacy-sensitive unpopular content related to one or more users. After compiling the list, the attacker finds the delay in accessing a cached content from the nearest router, also called hit time for the cache. Hit time can be calculated by requesting the same content twice. The first time, the content gets cached by the nearest router. The second time the request is satisfied by the cache of the nearest router. The delay in receiving the content second time is the hit time of the cache. After getting the hit time attacker requests privacy-sensitive contents one by one from the list to check whether it is cached or not. If the cache-hit occur then, the attacker can interpret that the user to which the attacker has linked the content while making the list of the privacy-sensitive unpopular contents must have accessed cache recently.

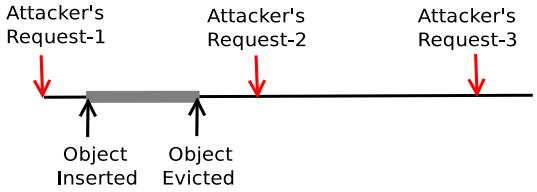

Fig. 3. Attacker probing at low frequency unable to detect request for content object

Algorithm 1 Measurement of t_c

OUTPUT: $t_{c,u}, t_{c,l}$

$N = MAX\_ITERATION$
$o(j) = /alice/generate/ < j >$ ▷ Object name template
for $j = 1; j \leqslant N; j = j+1$ **do**
  $request(o(j))$
  $t_{i,j} = currentTimeStamp()$
$t_{c,l} = 0$
$t_{c,u} = \infty$
for $j = 1; j \leqslant N; j = j+1$ **do**
  $request(o(j))$
  if $request(o(j)) = CACHE\_HIT$ **then**
    $t_{c,l} = current$
    $sleep for \frac{1}{fm} seconds$
return $[t_{c,l}; t_{u,l}]$

4.2. Object Discovery Attack

In this attack, an attacker sends the root (represented as "/") namespace as the first request packet to the router. The router responds by sending any data packet which is cached. Again the attacker sends another request packet excluding the data packet that the attacker got earlier by setting the exclude field of interest packet to the name of the data packet received earlier. This time the attacker gets a new data packet as a reply. By repeating this action repeat-

edly, an attacker can know the configuration of data packets in CS.

This paper focuses on timing based attack. The object discovery attack utilizes NDN features such as exclude field so it can be mitigated by just disabling the exclude field. This may result in degradation of NDN performance. However, the timing-based attack can be made without utilizing any additional NDN interest. Also object discovery attack is not a well-focused attack as the attacker may have to wait for the the appropriate contents which are privacy-sensitive but in the timing-based attack, the attack can target intended content.

5. Proposed Countermeasure

We utilize Lauinger et al.[9] approach of performing the attack by measuring the average time for which a data packet exists in the CS called Characteristic Time (CT) represented as t_c. This t_c is used for setting the lower bound for the probing frequency. The probing frequency f_p should be greater than or equal to $\frac{1}{t_c}$ to achieve 100% detection.

5.1. Attack Model

For performing the attack, we have considered linear topology as shown in Fig. 2. The attacker and the consumer are attached to R1, and the producer is attached to R5. The attacker wants to check whether consumer recently requested privacy sensitive content published by the publisher who is publishing content for prefix "/prefix."

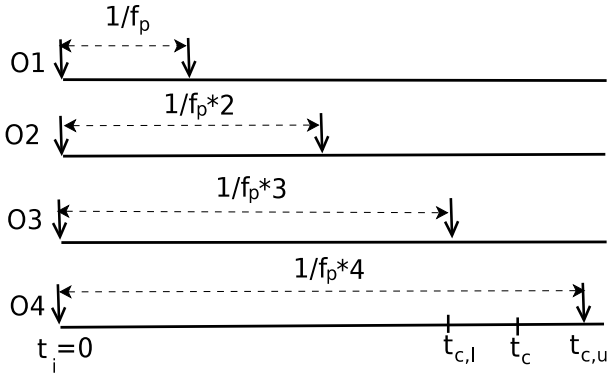

Fig. 4. Computation of CT using parallel probing

Algorithm 2 Parallel Cache Probing

$N = MAX\_ITERATION$
$c(j) = /alice/generate/ < j >$ ▷ Chunk name template
for $j = 1; j \leqslant N; j = j+1$ **do**
  if $request(c(j \bmod (m-1))) = CACHE\_HIT$ **then**
    "Some one requested O"
  else
    "No content is detected"
  $sleepfor \frac{1}{f_m} seconds$

We assume that there may be other consumer attach to R1 which can request content from any publisher. To make scenario easy to understand, we have shown only one consumer, and one publisher whose prefix is used by the attacker to check whether content belongs to that prefix is cached or not. We assume that the attacker can only access CS by sending interest packet. We also assume that the attacker can measure the time when the data is sent and received. We use LRU as cache replacement policy for CS.

Table 1: Detail of History data structure

Field	Description
name	First component of received interest name
time	Time at which the interest arrived
face	Interface at which the interest in received
count	Count number of successive malicious interests
isAttack	Set to true on attack detection
CC	Counts how much times the mitigation has been applied
AT	Attack Threshold
CountT	Counter Threshold
δ	Amount of deviation of successive interest from CT
Δ	Increment counter count so that counter decreases gradually

5.2. Measure Characteristic Time

For performing an attack, the attacker has to compute the probing frequency f_p. If f_p is too low, then a request from a user can go unnoticed by an attacker as shown in Fig. 3. Therefore for 100% detection $f_p \geqslant \frac{1}{t_c}$, here t_c is the time interval for which an object remain in cache [9]. Also, if f_p is too high, then the attacker may access the content cached by himself which may result in false detection. To avoid both the above scenario the value of f_p should be

almost equal to $\frac{1}{t_c}$.

Algorithm 3 OnInterest (Trigger on interest packet retrieval)

INPUT: interest, inFace
1: $firstComp = interest.get(0)$
2: $found = search(firstComp, inFcae)$
3: **if** $found == NULL$ **then**
4: $his = Create < Object > (History)$
5: $his.name = firstComp$
6: $his.time = now()$
7: $his.face = inFace$
8: $his.count = 1$
9: $his.isAttack = false$
10: $his.CC = 0$
11: $history.push(his)$
12: **else**
13: $it = history.advance(found)$
14: **if** $((avgLT - \delta) < (it.time - now()))\&((it.time - now()) < (avgLT + \delta))$ **then**
15: $it.count = it.count + 1$
16: $it.time = now()$
17: **if** $(it.count = AT)\&(it.isAttack = false)$ **then**
18: $it.isAttack = true$
19: $it.CC = it.CC + 1$
20: **if** $((it.isAttack = true)\&(it.CC < CountT))$ **then**
21: $it.CC = it.CC + 1$
22: **if** $(it.CC = CountT)$ **then**
23: $history.remove(it)$
24: **else**
25: **if** $(it.isAttack = true)$ **then**
26: $it.CC = it.CC + \Delta$
27: **if** $(it.CC \geqslant CountT)$ **then**
28: $history.remove(it)$
29: $Content = ContentStore.Lookup(interest)$
30: **if** $(Content! = NULL)$ & $(found == true)$ & $(it.isAttack == true)$ **then**
31: Delay data packet
32: Usual interest process

Lauinger et al.[9] have given Algorithm 1 for computing f_p. Initially, a large number of different contents are requested at the same time so that they get cached in the CS. The attacker requests each content after $\frac{1}{f_p}$ time interval. Thus first object is requested after $\frac{1}{f_p}$ time, the second object is requested after $\frac{1}{f_p} * 2$ time, ... the n^{th} object is requested after $\frac{1}{f_p} * n$ time and so on. Fig. 4 shows how an attacker can compute upper ($t_{c,u}$) and lower bound ($t_{c,l}$) of CT.

5.3. Perform Cache Privacy Attack

Lauinger et al. has given an approach to perform attack so that the request of attacker himself do not distract attack. The attacker requests chunks $c(0), c(1), ...c(m)$ of the same content parallely with frequency $f_p \geqslant \frac{1}{t_c}$ as shown in Algorithm 2. Cache hit implies that the given content is requested $\frac{1}{t_c}$ time unit before.

Table 2: Variation of CT in seconds for different values of frequencies and CS sizes

	CS size			
Frequency	**100**	**200**	**400**	**800**
100	3.2130	7.0670	16.7350	19.9760
200	1.6760	3.5780	8.2160	10.9540
400	0.665	1.842	4.2460	5.9930
800	0.171	0.9259	2.182	9.0930

5.4. Attack Detection and Mitigation

We have proposed an approach which utilizes the attacker's pattern for the detection of the CPA. For performing the attack, the attacker has to send a request after every $\frac{1}{t_c}$ time approximately. The occurrence of this pattern for a threshold number of times can be utilized to detect the attack. For detecting this pattern, we have used the History data structure implemented as a linked list that has six fields as given in Table 1. The attack detection is described in Algorithm 3. On receiving an interest packet, the router first checks if the first name component of the interest packet is in the History data structure or not. If the data is not found then a new entry is created having *name* field equals the name of the first component of the received interest packet and *time* field equals to the current time. The *face* field is set to the interface on which interest packet is received, *count* field is set to one, *isAttack* field is set to *false*, and Counter Count (*CC*) field is set to zero. If this entry is already present in History, then the router checks if the time difference between the current time and the time in the *time* field approximately equals to the average lifetime (*avgLT*) of the data packet or not. If yes then the value of *count* field is incremented and *time* field is set to current time. Here this approximate depends on δ which is a configurable value that can be set by the router according to convenience.

Next, the *count* field is checked if it equals to Attack Threshold (*AT*) then *isAttack* field is set to *true* and the *CC* field is incremented by one. Next, the *isAttack* field is checked if it is *true* and *CC* field is less than the counter threshold(*CountT*) then the

value of *CC* field is incremented by one. If the value of the *CC* field becomes equals to *CountT* then the corresponding entry is removed. In case the request does not lie within the detection zone and *isAttack* field is *true* then the *CC* field is incremented by the Δ threshold, this is done to ensure that the entry will be removed after some successful reception of the data packet. If the value of *CC* field becomes greater than or equals to *CountT* then the corresponding entry is removed. After detecting the attacker's pattern and updating entry of History data structure the data packet corresponding to interest packet is searched in CS. If the content is present in CS (in case of cache hit), its entry present in History, *isAttack* field is true then delay is applied. After applying the delay, the interest packet is processed according to the normal NDN forwarding pipeline.

6. Experimental Results

We have used ns-3 based simulator ndnSIM for the analysis of our approach. As shown in Fig. 2, linear topology is used for the evaluation of proposed countermeasure. We have taken fix PIT size of 12000kb. CS size varies from 100 to 800. Two consumer applications are installed on the consumer node; these are active during 0 to 600 seconds of simulation time. These applications requests interest packets whose names follow the Zipf distribution[2] with the value of q and s set to default of 0.7. The first application sends interest packets at a fixed rate of one packet per 10 seconds to producer application installed at producer node which listens to prefix "/p1." The second application sends packets at variable rates (per second), *i.e.*, 100, 200, 400, and 800 to producer application installed at producer node which listens to prefix "/p0." The attacker application is installed on attacker node which sends interest packet for prefix "/p1" with frequency equals to reciprocal of CT computed by the attacker.

Evaluation can be divided into three phases *i.e*, computing CT of the access router, performing attack at probing frequency which is calculated using CT, and countermeasure. For computing CT, we installed m additional consumer applications to the attacker node which sends a single packet to producer application installed on producer node listens to prefix "/qi" where i varies from 1 to m. First, these applications send a single interest packet so that CS of the access router caches data packets corresponding to these interest packets. Then after some time, these applications send interest packets according to Algorithm 1. The value of CT calculated for the different scenario is shown in Table 2. After calculating the CT, the attacker can perform the attack with frequency equals to reciprocal of CT.

We have considered five different scenarios for the evaluation of our countermeasure.

1. **No privacy**: We do not apply any mechanism to counter CPA.
2. **Uniform random delay**: We apply additional delay when the data item is satisfied by the CS this delay is obtained using the uniform random variable.
3. **Exponential random delay**: We apply additional delay when the data item is satisfied by the CS this delay is obtained using the exponential random variable.
4. **Our approach with uniform random delay**: This is our countermeasure approach in which the delay is obtained using the uniform random variable.
5. **Our approach with exponential random delay**: This is our countermeasure approach in which the delay is obtained using the exponential random variable.

Fig. 5 and Fig. 6 show the performance of our countermeasure as frequency and CS size changes in case of uniform random delay and exponential random delay respectively. For the evaluation of our approach, we have chosen the average delay of all the flows in the network per second. As we increase the frequency of the consumer, the rate of arrival of the data packets also increases which leads to the decrement of CT. The value of CT increases with the increase in the size of CS as shown in Table 2. We can also observe the decrease in average delay as the size of CS is increased and the increase in average delay as the frequency increases. We can also observe that

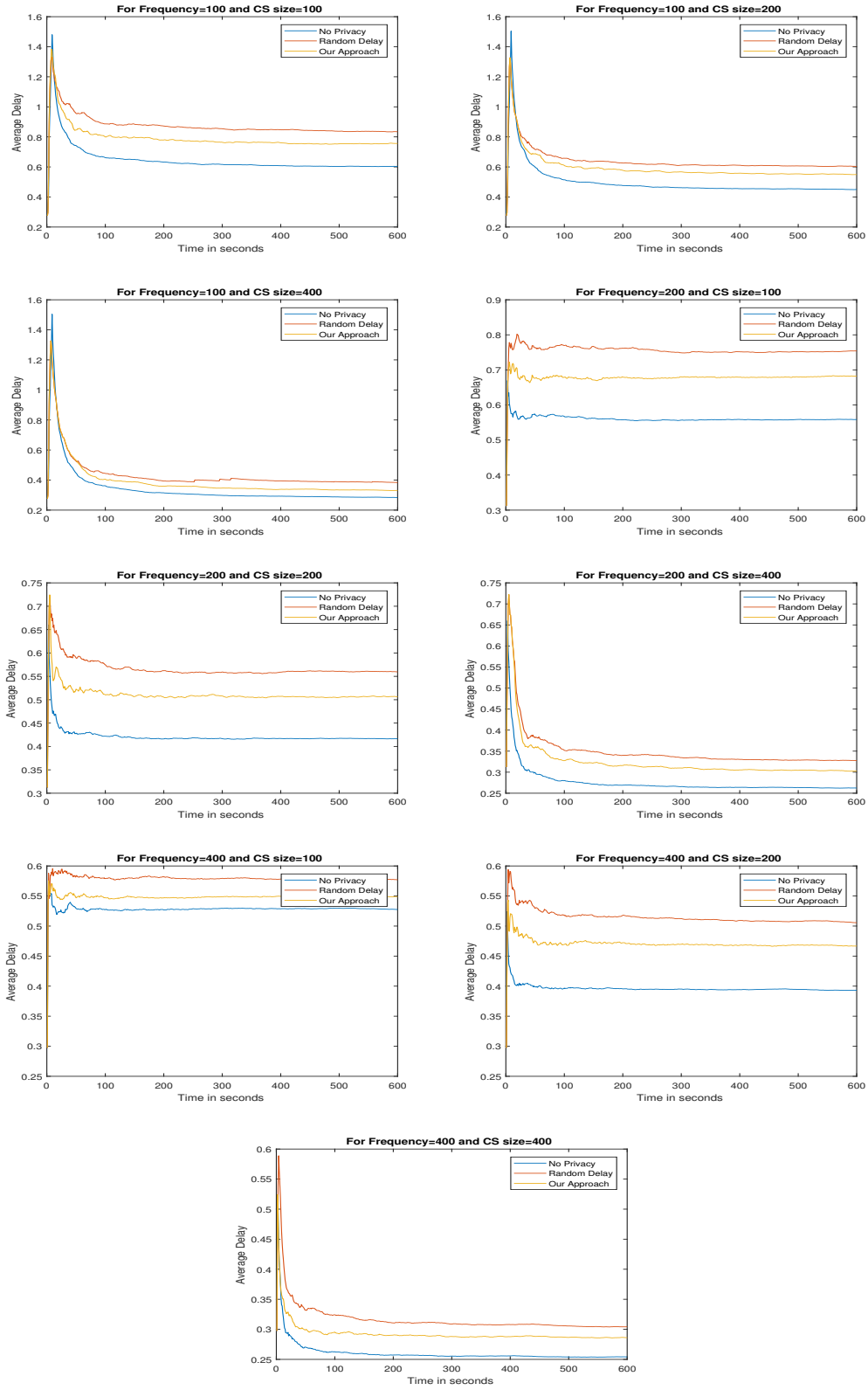

Fig. 5. Variation of average delay w.r.t. time when uniform random delay is applied

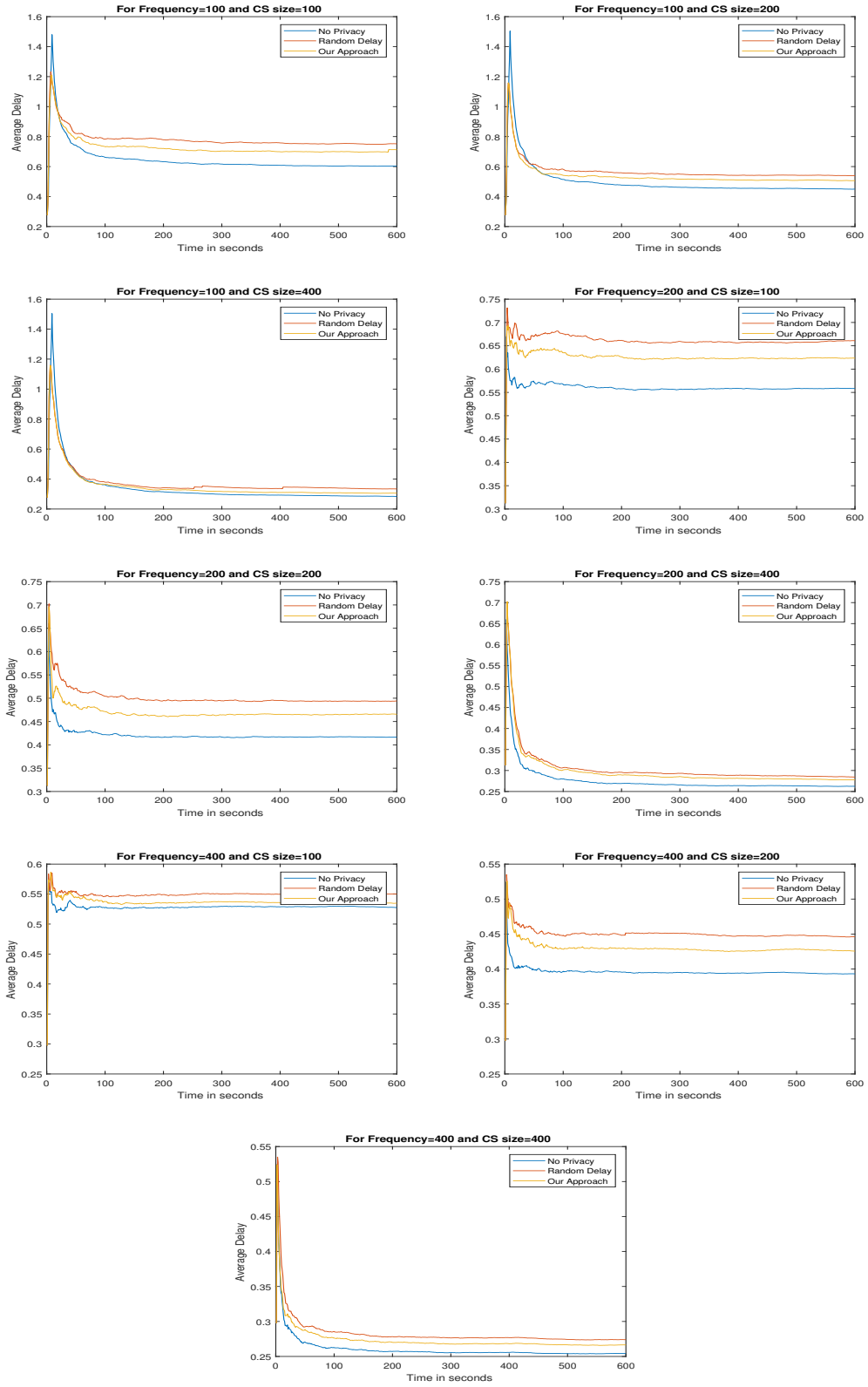

Fig. 6. Variation of average delay w.r.t. time when exponential random delay is applied

the best performance is achieved when no privacy is applied, this is the upper bound of the performance of the CS, but it does not have any privacy. In the random delay approach, the attacker would not be able to decide whether a request is served from the cache or by the producer, but the performance still degrades considerably. Our approach is better than the random delay approach.

For the uniform random delay, the range of values lie between 0 to 3000 ms having mean equals to 1500ms. For the exponential random delay, the input parameters are mean and upper bound which are equals to 1500ms and 6000ms respectively. From Fig. 5 and Fig. 6, we can observe that the average delay in case of exponential random delay is less than uniform random delay. Therefore approach based on exponential random delay performance better than uniform random delay.

7. Conclusion and Future Work

An approach to perform CPA efficiently is implemented by calculating CT and probing interest packet with the frequency equal to or higher than the reciprocal of CT. Then a detection scheme has been proposed for this attack and the countermeasure is applied through delay induction. The proposed approach is better than the approach which applies delay after each cache hit in terms of average delay. Exponential random delay performs better than uniform random delay.

However, the proposed work have been experimented for a single consumer whose request follow Zipf distribution in the linear topology. In a more complex scenario comprising of heavy traffic, it is difficult to estimate correct CT for the attacker. The value of the threshold can be optimized as a future task by evaluation of the approach under a different scenario.

References

1. Gergely Acs, Mauro Conti, Paolo Gasti, Cesar Ghali, and Gene Tsudik. Cache privacy in named-data networking. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pages 41–51. IEEE, 2013.
2. Lada A Adamic and Bernardo A Huberman. Zipf's law and the internet. *Glottometrics*, 3(1):143–150, 2002.
3. Alexander Afanasyev, Ilya Moiseenko, Lixia Zhang, et al. ndnsim: Ndn simulator for ns-3. *University of California, Los Angeles, Tech. Rep*, 4, 2012.
4. Ertugrul Dogruluk, Antonio Costa, and Joaquim Macedo. Evaluating privacy attacks in named data network. In *Computers and Communication (ISCC), 2016 IEEE Symposium on*, pages 1251–1256. IEEE, 2016.
5. Ali Ghodsi, Scott Shenker, Teemu Koponen, Ankit Singla, Barath Raghavan, and James Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM, 2011.
6. Cisco Visual Networking Index. Global mobile data traffic forecast update, 2016–2021 white paper, 2017.
7. Van Jacobson, M Mosko, D Smetters, and JJ Garcia-Luna-Aceves. Content centric networking. *whitepaper 2007*, 2009.
8. Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 181–192. ACM, 2007.
9. Tobias Lauinger, Nikolaos Laoutaris, Pablo Rodriguez, Thorsten Strufe, Ernst Biersack, and Engin Kirda. Privacy implications of ubiquitous caching in named data networking architectures. *Technical Report TR-iSecLab-0812-001, ISecLab, Tech. Rep.*, 2012.
10. Abedelaziz Mohaisen, Xinwen Zhang, Max Schuchard, Haiyong Xie, and Yongdae Kim. Protecting access privacy of cached contents in information centric networks. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 173–178. ACM, 2013.
11. Aziz Mohaisen, Hesham Mekky, Xinwen Zhang, Haiyong Xie, and Yongdae Kim. Timing attacks on access privacy in information centric networks and countermeasures. *IEEE Transactions on Dependable and Secure Computing*, 12(6):675–687, 2015.
12. Nonhlanhla Ntuli and Sunyoung Han. Detecting router cache snooping in named data networking. In *ICT Convergence (ICTC), 2012 International Conference on*, pages 714–718. IEEE, 2012.
13. George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.
14. FJ Ramón Salguero. Content mediator architecture for content-aware networks. *COMET EU FP7 Report*, 2010.

15. Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Patrick Crowley, Christos Papadopoulos, Lan Wang, Beichuan Zhang, et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.
16. Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, James D Thornton, Diana K Smetters, Beichuan Zhang, Gene Tsudik, Dan Massey, Christos Papadopoulos, et al. Named data networking (ndn) project. *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.
17. Zengwu Zhang and Ke Zhang. Research on security and privacy issues of ndn. 2014.