POSTER: Protecting Access Privacy of Cached Contents in Information Centric Networks[*]

Abedelaziz Mohaisen
Verisign Labs, VA, USA

Xinwen Zhang
Huawei Technologies, CA, USA

Max Schuchard
University of Minnesota, MN, USA

Haiyong Xie
Huawei Technologies and USTC, China

Yongdae Kim
KAIST, Daejeon, South Korea

ABSTRACT

In information centric network (ICN), contents are fetched by their names from caches deployed in the network or from origin servers. Once the contents are fetched from the origin server, it is replicated and cached in all routers along the routing and forwarding paths from the user that issues the interest to the origin server, thus allowing further "interests" by other users to be fulfilled quickly. However, the way ICN caching and interest fulfillment work pose a great privacy risk; the time difference between response for interest of cached and uncached contents can be used as an indicator to infer whether or not a near-by user previously requested the same contents requested by the adversary. This work introduces the extent to which the problem is applicable in ICN and provides several solutions to address it.

Categories and Subject Descriptors: C.2.0 COMPUTER COMMUNICATION NETWORKS: Security and protection
Keywords: Information centric networks, Privacy, Caching.

1. INTRODUCTION

In information centric networks (ICN) [1], contents are fetched by their names from caches deployed in the network or from origin servers—servers that serve contents if they are not cached in the network. Once contents are fetched from an origin server, they are replicated and cached in all routers along the routing and forwarding paths from the user that issues the interest to the origin server, thus allowing further interests to be fulfilled quickly. For example, when another user issues an interest in these contents that have been previously served to a user on the same path, the interest is fulfilled from the near-by cache. However, the way ICN interest fulfillments work pose a great privacy risk. In particular, the time difference between response for interest of cached and uncached contents can be used as a side channel to infer whether a near-by user previously requested the same contents as the adversary.

Consider the topology in Figure 1, which depicts two users U_1 and U_2, and a set of routers r_0 to r_4 (each has own cache) connecting both users to an origin server that holds some contents. Suppose that user U_2 is the adversary, whereas user U_1 is honest. If U_1 issues an interest in contents N that reside behind r_0, the interest should traverse the path $U_1 \to r_3 \to r_2 \to r_1 \to r_0$ from which it retrieves the contents requested. The contents are then sent back over the returning path $r_0 \to r_1 \to r_2 \to r3 \to U_1$. In total, the path from U_1 to the source of the contents and the returning path to U_1 have four hops each. The total round trip time required

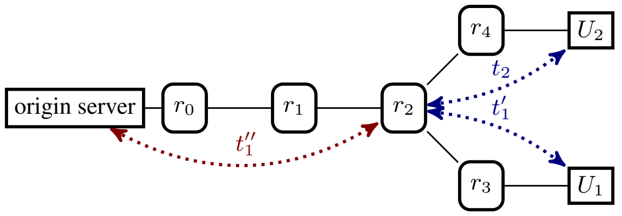

Figure 1: Toy example of the timing attack. $t_1 = t'_1 + t''_1$

for sending the request until starting to receive contents on the returning path is t_1. On the other hand, if U_2 is to request the same contents by its name, N, the path that the interest would traverse is $U_2 \to r_4 \to r_2$, and the contents would return on the reversed path ($r_2 \to r_4 \to U_2$), which is two-hop in each direction, and would require a time t_2. Obviously, the time t_1 is a greater than t_2, which an adversary, like the user U_2, can use to infer that user U_1 has accessed the contents N.

Although pinpointing U_1 in real settings precisely may require additional side information [3], an attack like the one described above—which finds out a 1-hop away user without naming him—is still critical since it reduces the anonymity set of that user greatly.

In this work, we examine the timing attacks on privacy in ICN caching and propose three solutions that come at varying levels of costs and complexity. We rely on randomly generated time paddings to disguise responses for interests issued from the same domain, thus increasing the anonymity set of privacy-related contents' interest issuer. While we disclaim the novelty of the attack—shown in other context in [4], we are the first to observe its inherent applicability to ICN and to provide solutions and mitigations.

2. PRELIMINARIES AND TERMINOLOGY

In ICN, contents are fetched by their names [1]. An ICN consists of routers, where each router has a cache, and edge routers are connected to users and origin servers. *Interest* in ICN encapsulate requests for contents by their names. An *origin server* is a server that originates contents to be served in the network, thus fulfilling interests. The contents may or may not be cached in the network. In the rest of this work, we use Round Trip Time (RTT) to denote the time from the start of sending an interest until the start of receiving contents fulfilling it. In ICN, contents are forwarded back to users on the same path they are requested by that user, thus PIT (*pending interest table*) at each ICN router records which interest is not fulfilled yet. A *face* in ICN is the port at which data is sent or received in the router. In one of our protocols we make use of an access point (AP), which is the closest connecting point of the user to the ICN (not to be confused with wireless access point). flist and ulist are lists of faces and users, while rtimes is a list of the number of times the contents are fetched by each (either face or user). pmode is a flag to indicate that the privacy of the contents being accessed need to be preserved in future access and requests.

[*]Haiyong Xie is supported in part by the NSFC Grant No. 61073192 and by the 973 Program Grant No. 2011CB302905.

3. PROTECTION MECHANISMS

Simple solutions cannot prevent the attack, although they can greatly degrade the benefits of ICN. For example, if the edge router always generates an equal delay to the RTT from that router to the origin sever, two in-row requests by an adversary would reveal whether the requested contents are flagged private or not by others.

The first technique to address the attack enables each user concerned about the privacy of his access to contents to use a privacy mode, upon which the edge router through which the interest is served (or propagated to the network) maintain a state of the user, the requested contents' name, and the number of times the user requested it. When other users request the same contents for the first time, the router generates random delay to simulate a network delay after which sends the contents to the requester. This technique requires keeping states users requesting privacy-related contents and their times of requests, which represents an overhead. On the other hand, this solution can be tuned to maintain shorter RTT as in ICN.

To reduce overhead at edge routers and to enable inter-domain privacy, we let routers only keep states for requests coming on faces. When an interest of cached contents arrives for the first time at a certain face, the edge router generates random delay and serves the contents so that to preserve the privacy of other users in other domains, who requested the contents before. When a face has previous requests, the contents are served to the requester immediately. Although this technique reduces the overhead of the first technique, it does not enable intra-domain privacy preservation.

To enable low granularity of the privacy and to reduce the overhead at edge routers, we maintain the same states of users as in first solution but in the access points. We then use these states to collaboratively tell routers if contents have been requested before by the same user or not. In the following we explain the three protocols.

Before going into the details of the protocols, we first introduce the time (delay) generation procedure. The procedure is performed by an edge router, and takes several parameters based on the specific protocol in which it used to generate td, the number of hops to be added as noise to prevent the timing attack. Particularly, for a content name $n \in N$, the total number of hops h, RTT td_x, and the time delay for the first hop td_0, $td(n)$ is chosen as follows; for a given n, the same value of $td(n)$ is used for subsequent requests.

$$td(n) = \begin{cases} 0 & h = 1 \\ 2td_0 < td(n) < td_x & h > 1 \end{cases} \qquad (1)$$

3.1 The "Vanilla" Approach

The protocol is described in algorithm 1. The main ingredient of the protocol is a carefully chosen delay added to subsequent responses to make them similar to responses that fall back on the origin servers to ensure that the contents that are sent to an adversary do not expose timing pattern. For that, the protocol relies on states stored by each router. Particularly, for a user u (U_1 in Figure 1), its edge router (r_2 in Figure 1) maintains $\varphi(u,n) : U \times N \to INT$, where U, N, and INT are the sets of users, content names, and integers, respectively. $\varphi(u,n)$ indicates the number of times that user u has accessed the content name n.

3.2 An Efficient Approach

This protocol is in algorithm 2. The main idea of this protocol is to reduce the states stored in each router to that of faces and the number of requests that have been served to users over each face, rather than maintaining a large number of states per user. The main observation made in this protocol is that interests from different domains (or sub-domains) traverse different faces at the edge router while interests coming from the same domain (or sub-domain) would traverse the same face at the edge router. To that end, states of faces are stored and maintained in each router, and decisions to preserve privacy of access are made upon those states.

Algorithm 1: The "vanilla" approach.

```
Input: $n$ - a content name, $u$ - a user, $\varphi$ - access state,
       $Ints = (u, n, \text{pmode}, ts_0)$
Output: A data packet to $u$ in a privacy-preserving manner.
1  When $R$ receives $Ints$ from $u$, it records $ts_1$, the timestamp of
   interest arrival, and computes $td_0 = ts_1 - ts_0$ as a one-hop
   time delay.
2  if pmode == 0 then
3      if $td(n) == 0$ then
4          // default value $td(n) = 0$
5          $R$ follows ICN protocol to obtain $Data$;
6          $R$ returns $Data$ to $u$;
7      else
8          $R$ follows ICN protocol to obtain data packet $Data$;
9          $R$ delays $td(n)$;
10         $R$ returns $Data$ to $u$;
11     end
12 else
13     if $\varphi(u,n) == 0$ then
14         $R$ follows the ICN protocol to obtain $Data$;
15         $R$ records $ts_2$ upon the arrival of $Data$, and computes:
16            $td_x = ts_2 - ts_1$; // RTT from $R$ to origin server
17            $h = td_x/(2td_0) + 1$; // expected # of hops from $u$
           to the origin server
18            Generate $td(n)$ according to Eq. 1;
19            $\varphi(u,n)++$;
20         $R$ returns retrieved $Data$ to $u$;
21     else
22         $R$ returns cached $Data$ to $u$;
23     end
24 end
```

Unlike Algorithm 1, each router stores $\varrho : F \times N \to INT$, where F is the set of faces. $\varrho(f,n)$ indicates the number of times that content name n has been requested from face f.

3.3 Low Granularity Approach

The main limitation of the approach in §3.2 is that it does not enable low granularity of the preserved privacy when both the adversary and honest user are using the same AP, unlike the protocol described in §3.1. To enable that, in the protocol in §3.1 we maintain several states in the router, which is a high overhead that can be misused whereas the protocol in §3.2 reduces this overhead greatly. The protocol in algorithm 3 aims to maintain the advantage of both protocols by maintaining and distributing these states concerning access pattern of individual users at the APs.

The main idea of the protocol is to distribute states of users sending interests from within a certain domain on the APs associated with these users, where decisions for access privacy are made at the router with the help of the AP. The protocol assumes that faces' states are maintained in the routers and the users states are in the APs. We keep in mind that an AP is the closest connecting point of the users to routers (e.g., between U_1 and r_3 in Figure 1).

4. INITIAL RESULTS AND ANALYSIS

To understand the potential of the attack proposed in this work in reality and how our designs impact the performance of ICN, we instrument the CCNx simulator (`https://www.ccnx.org/`) with real-world per-hop round trip delays when issuing interests from within our campus in Santa Clara, CA, to each of the Alexa top-100 sites. We use traceroute (`http://www.traceroute.org/`) to obtain the hop count and per-hop RTT delay to each site

Algorithm 2: An efficient approach.

```
Input: $n$ - content name, $f$ - face id, $\varrho$ - access state,
       $Ints = (n, \mathsf{pmode}, ts_0)$
Output: A data packet to $f$ in a privacy preserving manner.
1  When $R$ receives $Ints$ from an access point AP through face $f$,
   it records $ts_1$, the timestamp of interest arrival, and computes
   $td_0 = ts_1 - ts_0$ as a one-hop time delay.
2  if $n$ is not in $R$'s cache then
3      $R$ follows the ICN protocol to obtain data packet $Data$
       from the origin server;
4      $R$ records $ts_2$ upon the arrival of $Data$, and computes:
5          $td_x = ts_2 - ts_1$; // RTT from $R$ to origin server
6          $h = td_x/(2td_0) + 1$; // expected # of hops
7          Generate $td(n)$ according to Eq. 1;
8          $\varrho(f, n) + +$;
9      $R$ returns $Data$ to AP via $f$;
10 else
11     if $\varrho(f, n) == 0$ then
12         $R$ generates $td(n)$ as in Eq. 1;
13         $R$ delays $td(n)$.
14         $R$ returns $Data$ to the AP via $f$;
15     end
16 end
```

(origin server), and feed them to a dummy CCNx topology corresponding to the toy example in Figure 1. To unify our analysis and discussion, we limit our attention to 24 sites that have exactly 16 returned valid hops in traceroute (15 hops outside of our campus). A boxplot of the RTT up to each hop (1 to 15, until reaching the origin server) as a ratio of the total RTT to the origin server is shown in Figure 2 (more details on characteristics of this data are in [2]).

First, we examine whether an adversary co-located one-hop away from a legitimate user will be able to exploit the timing attack explained earlier to infer whether some contents are being retrieved by that user or not. We note that as per the ICN caching policy, contents are replicated and cached at each hop, thus future requests are fulfilled immediately from the closest router to the user. From Figure 2, we observe that an adversary who is co-located with the user who requested these sites will benefit from the caching, and would ideally reduce the total RTT for fulfilling a request by a cache hit at the first hop by around 98% for the most conservative site (and more than 99% for the average site). Even when a cache-miss happens, an RTT by a cache hit at the fourth hop away from the user, for example, would be 40 times at average (and 25 times at worst) less than the RTT when retrieving contents directly from the origin server—although this scenario may not breach the privacy of users access pattern since a 4-hop network has a large anonymity set. By feeding the timing profiles of Figure 2 in CCNx we observe that the

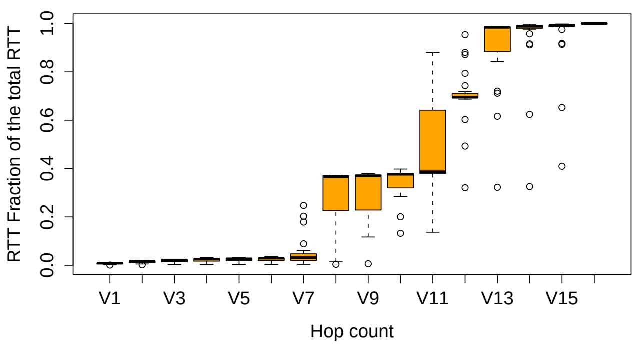

Figure 2: A boxplot of the RTT as a ratio of the total RTT for 24 sites in Alexa's top 100, with 16 hops to each site.

Algorithm 3: Low granularity approach.

```
Input: $n$ - content name, $f$ - face id, $u$ - user id, $\varrho$ - access
       state, $Ints = (n, \mathsf{pmode}, ts_0, flag = false)$
Output: Returns data packet to $u$ in a privacy preserving
        manner.
1  $u$ issues $Ints$ with pmode enabled for $n$. $u$ records $ts_0$ of $Ints$
   and sends them AP that connects $u$ to the ICN.
2  When the AP receives $Ints$:
3  if $\varphi(u, n) == 0$ then
4      AP discards the pmode tag and flags $Ints$ with
       $flag = true$;
5      AP forwards $Ints$ to router $R$;
6  else
7      AP forwards $Ints$ to router $R$;
8  end
9  Upon receiving $Ints$ from face $f$, the router $R$:
10 if $n$ is not in $R$'s cache then
11     $R$ follows the ICN protocol to retrieve the contents from
       the origin server and serve them to $u$.
12 else
13     if $\varrho(f, n) == 0$ then
14         $R$ generates $td(n)$ with Eq. 1;
15     else
16         if $flag == true$ then
17             $R$ fulfills the interest from cache
18         else
19             $R$ generates delay $td(n)$ as in Eq. 1;
20             $R$ delays response by $td(n)$;
21             $R$ returns cached content $n$;
22         end
23     end
24     $R$ delays $td(n)$;
25     $R$ returns $Data$ to face $f$;
26 end
```

network latency is the dominating part of the RTT in ICN, and other ICN-related delay is negligible. We conclude that an adversary that relies only on the timing information can easily and successfully infer that the contents are being cached in a near-by router.

Second, we look at how our designs impact the performance of ICN. One critical parameter for our designs is d, the number of hops that an edge router estimates and generates to use as timing noise. Even when the router has the capability to record a per-hop RTT and add them as noise, the overhead as additional time delay added to the RTT of fulfilling requests to users still maintains the benefits of ICN. For example, when $td = 6$ (which is one-third of the hop count to the origin server thus providing reasonable anonymity set), a request to an average site would be fulfilled about 40x faster than retrieving contents from the origin server. Even for sites with the longest RTT, that would be 25x faster than getting contents from the origin server—25% and 75% RTT sites fulfill requests at about 33x and 4x respectively for $td = 7$. As before, RTT is dominated by network latencies, whereas CCNx delays are negligible, supporting our claim that our designs maintain ICN's benefits.

5. REFERENCES

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. Braynard. Networking named content. In J. Liebeherr, G. Ventre, E. W. Biersack, and S. Keshav, editors, *CoNEXT*, pages 1–12. ACM, 2009.

[2] A. Mohaisen, X. Zhang, M. Schuchard, H. Xie, and Y. Kim. Protecting access privacy of cached contents in information centric networks. TR, UMN, 2012.

[3] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security*, pages 199–212, 2009.

[4] E. W. Felten and M. A. Schneider Timing Attacks on Web Privacy. In *ACM Conference on Computer and Communications Security*. 2000.