Protecting Access Privacy of Cached Contents in Information Centric Networks

Abedelaziz Mohaisen[†] Xinwen Zhang[‡] Max Schuchard[◇] Haiyong Xie[‡⋆] Yongdae Kim[♮]
[†]Verisign Labs, VA, USA [◇]University of Minnesota, MN, USA [‡]Huawei Technologies, CA, USA
[⋆]Univ. of Sci. & Tech. of China, China [♮]KAIST, South Korea

***Abstract*—In recently proposed information centric networks (ICN), a user issues "interest" packets to retrieve contents from network by names. Once fetched from origin servers, "data" packets are replicated and cached in all routers along routing and forwarding paths, thus allowing further interests from other users to be fulfilled quickly. However, the way ICN caching and interest fulfillment work poses a great privacy risk: the time difference between responses for an interest of cached and uncached content can be used as an indicator to infer whether or not a near-by user has previously requested the same content as that requested by an adversary. This work introduces the extent to which the problem is applicable in ICN and provides several solutions that try to strike a balance between their cost and benefits, and raise the bar for the adversary to apply such attack.**

***Index Terms*—Information centric networks, privacy, side channel attacks, caching.**

I. Introduction

Information centric networks (ICNs) have been proposed as new Internet architectures towards secure and efficient content dissemination. In several ICNs such as content centric network (CCN) [17] and named data network (NDN) [33], contents are fetched by their names from caches deployed in the network or from origin servers—servers that serve the contents if they are not cached in the network. In such ICN architectures, once a content data packet is fetched from an origin server, it is replicated and cached in all routers along the routing and forwarding path—starting from the router that connects user who issues the interest to the one that connects the origin server to the ICN—thus allowing further interests with the same content name to be fulfilled quickly [17]. 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. This design choice—as advocated by many ICN designs and architectures—is considered a great advantages in reducing overall content retrieval latency [17], [33].

However, this universal caching mechanism in ICN poses a great privacy risk. In particular, the time difference between data response for an interest of cached when compared to uncached content data packet can be used as a side channel to infer whether a near-by user has previously requested the same content or not. The following example illustrates the problem.

A. Example of Attack on Privacy in ICN

Consider the topology in Figure 1, which depicts users U_1 and U_2, and a set of routers r_0 to r_2 (each with its own cache)

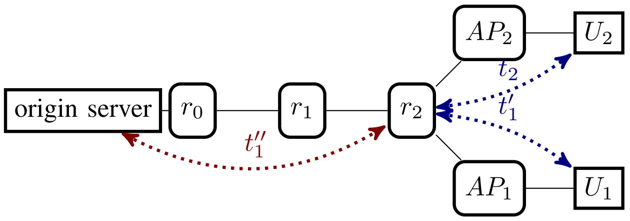

Fig. 1. Toy example of timing attack in ICN. $t_1 = t_1' + t_1''$. A user U_2 would be able to infer whether user U_1 has accessed a content based on the different in RTT for cached and uncached content. Notice that AP_1 and AP_2 are access points that connect user U_1 and U_2 to the ICN via router r_2.

connecting both users to an origin server that holds content with name n. Suppose that user U_2 is the adversary, whereas user U_1 is honest. If U_1 issues an interest in content n that resides behind r_0, the interest traverses the path $U_1 \to AP_1 \to r_2 \to r_1 \to r_0$, from which it retrieves a requested data packet of the content. The packet is then sent back over the returning path $r_0 \to r_1 \to r_2 \to AP_1 \to U_1$. In total, the path from U_1 to the source of the content and the returning path to U_1 have four-hop each. The total round trip time required for sending the request until starting to receive data packets on the returning path is t_1. On the other hand, if U_2 is to request the same content by its name, n, the path that the interest would traverse is $U_2 \to AP_2 \to r_2$, and the contents would return on the reversed path ($r_2 \to AP_2 \to U_2$), which is two-hop in each direction, and would require a time t_2. Obviously, the time t_1 is greater than t_2, which an adversary U_2 can use to infer that user U_1 has accessed the content n.

Although pinpointing U_1 precisely among many users who use the same domain and are within the same geographical proximity may require additional side information [22], an attack like the one described above—which finds out a 1-hop away from a user without naming him—is still critical since it reduces the anonymity set of that user greatly. Such scenario is equal in value to identifying individual users when combined with real-world inference applications to privacy. For example, an adversary in a business intelligence attack might be more interested in knowing what contents are being retrieved by a competing company than by individual users working for that company. This attack would be possible if the adversary is co-located with that company behind an edge router, and using the above technique.

B. Shortcomings of Simple Solutions

Simple solutions cannot prevent the problem exposed in this attack. For example, a user can flag a content object with a privacy value so as to enforce routers not cache data packets in the network of that particular contents at all time, for all subsequent interests (regardless to whether they are flagged or not). However, since what is considered of privacy value differs from a person to another, this implies an increased delay in delivering the content to other benign users, thus degrading the advantage of ICN. Also, such solution is always vulnerable to active attacks: an adversary who wants to infer some information about the content can do that by performing two consecutive requests on the same name. In the first request, and if the content is not previously flagged for privacy reasons by other users, the requested name would result in data packets being cached. The second request in that case will result in cache-hit, and the content will be served to the adversary quickly. On the other hand, if the content is previously flagged for privacy reason by other users, the second request would result in a delay close to the delay in the first request, from which the adversary can infer that such content is not cached in the network, and that another user has likely flagged such content for privacy reasons.

Other solutions to solve the problem by intelligently deciding caching based on how far a router is away from users requesting such contents do not work in practice as well. For example, if a router wants to decide caching based on how far a user is away, it has to either know the user's location in advance or have the location provided to it at the time of caching. The first approach requires partial knowledge of the topology that can go easily beyond the resources available in typical routers. The second approach is vulnerable to misuse by an active adversary—an adversary who wants to negatively impact the experience of other users can flag any content from an arbitrary location so as to make it never cached in near-by routers to end users.

C. Contributions

In this work, we examine timing attacks on privacy in the ICN caching mechanism, and propose three solutions that come at varying levels of cost and complexity. Our approach relies on randomly generated time paddings to disguise responses for interests issued from users in the same domain, thus increasing the anonymity set—set of domains that requests potentially come from—of privacy-concerned interest issuers. While we disclaim the novelty of the attack—shown in other context in [22], we are the first to observe its inherent applicability to ICN architectures and provide solutions and mitigations. The unique contribution of this work is as follows:

- We demonstrate timing attacks on the universal caching mechanism proposed in ICN designs like CCN and NDN. For that, we make use of fine-grain per-hop timing measurements of cached and uncached contents using real-world time measurements with CCNx, a prototype implementation of the CCN [26]. We disclaim the originality of the attack in its general form but claim its suitability and applicability to ICN as a novel contribution.
- We propose three protocols, each with different levels of complexity, cost, and privacy guarantees that prevent an adversary co-located with benign users to infer whether they have accessed certain contents or not by relying on the timing attacks. Each and every of these protocols tries to strike a balance between the privacy provided to legitimate users from potential adversary, and the overhead added for requests performed by other legitimate users to the privacy-related contents.

D. Organization

The organization of the rest of this paper is as follows. In Section II we review the preliminaries and terminologies used in this paper. In Section III we introduce three protocols to solve the problem and maintain the privacy of users access, where each protocol comes at different cost and privacy guarantees. In Section IV we present our simulation results to validate the attack and evaluate the performance of our defense protocols. In Section V we highlight several discussion points, including potential attacks and their applicability to our protocols. Section VI reviews related work and Section VII concludes this work and point out our future work.

II. Preliminaries and Terminology

In this section we first review the terminologies related to ICN architectures—with CCN and NDN in mind—in §II-A. We then review the attack model used in this paper in §II-B.

A. Terminologies and ICN Operations

In ICN, contents are fetched by their names [17]. An ICN consists of routers, where each router has a cache, and edge routers are connected to users and origin servers. An *Interest* in ICN encapsulates a request for a content packet by its name. An *origin server* is a server that originates contents to be served in the network, thus fulfilling interests. The contents (data packets) may or may not be cached in the network. In the rest of this work, we use total Round Trip Time (RTT) to denote the time from the start of sending the first interest until the start of receiving a content packet fulfilling it (also known in the literature as Time to First Byte; TTFB). Similarly, we define RTT per hop. In ICN, contents are forwarded back to a user on the same path as 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 a router. In 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 a wireless access point). Each router maintains a set of states to record the number of times that a privacy-sensitive content object has been fetched by each user or face. pmode is a flag to indicate that the privacy of a content name being accessed need to be preserved in future access and requests.

B. Attack Model

We consider an adversary co-located with an honest user who tries to access contents from ICN. To this end, we assume that the adversary has the capability to perform fine-grained time measurements to perform attacks. We also assume that the attacker has a list of potential "names", where he wants to verify whether the benign user has accessed such names or not. We do not assume any insider attacks, since such names are easy to infer given that domain-specific names are common among people working in that domain, and are easy to infer. From this assumption it follows that the adversary has no control over which path interests are sent, and cannot be geographically distributed to perform an intersection attacks by combining several measurements at different network locations (cf. Section V). Finally, for the operation of our attack, we assume that the adversary has enough time to perform the attack, which implies that the content caching lifetime is long enough that the adversary would have a cache hit for contents previously cached by the benign user's requests.

In this paper we assume that the underlying infrastructure used by both adversaries and benign users is honest. In particular, a common router that holds traffic of the adversary and the honest user cannot collude to perform an attack against the benign user (e.g., r_2 in Figure 1). On the other hand, the adversary, if at the scale of a subdomain, may control a router where no traffic of the benign user passes through (e.g., r_3 could replace AP_2 in Figure 1). This assumption can be further used by the adversary to enumerate in real-time what contents are being consumed by other users in his domain, and to help him improve the inference attack on other users within proximity but in other domains (see §I-A for such scenario).

The attack discussed in this paper is applicable to both CCN and NDN and to a lesser extent to other future Internet architecture proposals [5], [16], [24]. Notice that there have been several efforts in improving caching in ICN, and the main motivation of such designs is performance: each of these designs tries to improve network performance by reducing cache-miss. While our attack might be less applicable to these designs, they might be vulnerable to other cache-related attacks as in [30]. Verifying how vulnerable are these architectures to our attack is left as a future work.

III. Protection Mechanisms

As mentioned before, simple solutions cannot prevent the timing attacks for privacy while greatly degrade the benefits of ICN architectures. Also, intelligent caching requires a topology knowledge that is beyond a router's resourcers.To this end, we propose several solutions without requiring such knowledge.

A. Summary of the Protection Techniques

The first technique to address the attack, named the "vanilla" approach, enables each user concerned about the privacy of his access to use a *privacy mode*, and the edge router maintains a state of the user, the requested content names, and the number of times the user has requested them. When other users request the same contents for the first time, the router generates random delay to simulate a network delay before sends the contents to the requester. This technique requires keeping states: user ids, content names, and the times of requests, which represent necessary overhead in the edge router. On the other hand, this solution can be tuned to maintain shorter RTT as in ICN. The detailed protocol is introduced in Section III-B.

To reduce the overhead in the vanilla protocol but by enabling higher granularity of privacy, we let routers only keep states for requests coming on faces, and maintain per-face states instead of per-user states. In our "efficient" approach, when an interest of a cached content arrives for the first time at a certain face, the edge router generates random delay and serves the content so as to preserve the privacy of other users in other domains (their requests come from different faces), who have requested the contents before. When a face has previous requests for the same content, the content is served to the requester immediately. Although this technique reduces the overhead of the first technique, it does not enable low privacy preservation. The detailed protocol is in Section III-C.

In order to enable low granularity of the privacy and to reduce the overhead at edge routers, we maintain the same states of users as in the vanilla approach but in access points. We then use these states to collaboratively indicate routers if the target contents have been requested before by the same user or not. In particular, when a request is issued by a user setting behind an access point (AP), the access point maintains his identifier, and the number of times he has previously requested the content name. If this is for the first time, and the content is previously marked for privacy-related query, and the request of that content is flagged. Accordingly, the router generates random delay as before, and then serves the content to the user via the AP. If the content is privacy-related, and the user has already requested the content before, the request is not flagged, and the contents are served directly to the user. If the content is flagged, or not flagged, but not cached, it is served according to the original ICN protocol. This protocol maintains the privacy of the requester from the same domain, while reducing the states stored on the router for faces statistics, whereas all user statistics are stored on the close by AP. The detailed protocol is in Section III-D.

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 (from the user to the edge router), $td(n)$ is chosen as follows to balance privacy and the degradation of service as compared to fetching contents directly from the cache. For a given n, the same value of $td(n)$ is used for subsequent requests.

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

B. The "Vanilla" Approach

The vanilla algorithm to prevent timing attacks on privacy in ICN is described in Algorithm 1 and illustrated in Figure 2. The main ingredient of the algorithm is a carefully chosen

Algorithm 1: The "vanilla" approach to preserving the privacy of cache access. The description makes use of the toy example in Figure 1 and is illustrated in Figure 2.

```
Input: $n$ - a content name, $u$ - a user, $\varphi$ - access state,
       $Ints = (u, n, \mathsf{pmode}, ts_0)$
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 packet $Data$
           from the origin server;
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 packet
           $Data$ from the origin server;
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
```

Algorithm 2: The efficient approach to preserving the privacy of cache access. The description makes use of the toy example in Figure 1 and is illustrated in Figure 2.

```
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 from $f$ to
       the origin server
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
```

delay added to subsequent responses to make them similar to the responses that fall back on the origin servers to ensure that the contents that are sent to an adversary do not expose timing patterns—such patterns could be used to infer if other users have requested the same contents. For that, the protocol relies on states stored by each edge router to name the contents that are of privacy-value to users, the number of times the contents are being served to each user, and the user id.

Particularly, for a user u (U_1 in Figure 1), its edge router (r_2 in Figure 1) maintains $\varphi(u, n) : U \times N \rightarrow INT$, 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. At the beginning, assuming benign user U_1 first generates interest $Ints = (U_1, n, \mathsf{pmode}, ts_0)$ with $\mathsf{pmode} = 1$, where ts_0 is the timestamp of when the interest is issued. When r_2 receives this, it follows the ICN protocol [17] to retrieve a data packet $Data$ from the origin server, and records ts_2 upon the arrival of the first packet in response of the interest. Following Eq. 1, r_2 computes expected number of hops from the user U_1 to the origin server as $h = td_x(N)/(2td_0) + 1$, and then records td_x along with (U_1, n), and updates the φ to indicate the times that the user has accessed the content. r_2 then serves the content to U_1. When another interest for n is issued by user U_2, who is a potential attacker, the router r_2 acts in response to this interest as follows: If U_2 has previously requested n, r_2 responses directly and serves contents from the cache. Else r_2 applies the random delay and returns $Data$ to U_2.

C. An Efficient Approach

While the vanilla algorithm preserves the privacy of user's access history from attackers in the same domain, it consumes significant resources in edge routers, especially when threats from different domains are concerned, where each domain may have large number of users. In order reduce the states stored in each router, a more efficient way is to maintain per-face state instead of per-user ones. The main observation made here is that interests from different (sub-)domains traverse different faces at an edge router, while interests coming from same (sub-)domain would traverse the same face. Accordingly, per-face

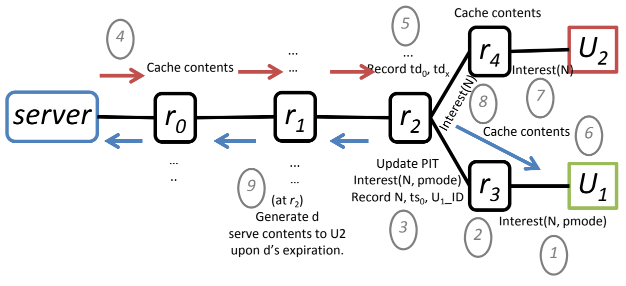

Fig. 2. An illustration of the protocols in Algorithm 1-3. State updates as well as repeated steps (at r_1 and r_0) are omitted for brevity. Notice that r_3 and r_4 are the access points of both users in Figure 1.

states are stored and maintained in each router, and decisions to preserve privacy are made upon those states.

Algorithm 2 shows the protocol for an edge router. Unlike the protocol in Algorithm 1, each router stores $\varrho : F \times N \rightarrow INT$, 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. The protocol can be illustrated in Figure 2, where router r_2, for example, keeps track of the faces connecting it to other routers and access points (e.g., r_3 and r_4), and the times each face has requested content names that have been previously marked as privacy-related contents. After that, r_2 follows the protocol by adding random delays when fulfilling interests that could potential thwart the privacy of other users' access.

D. Low Granularity Approach

The main shortcoming of the approach described in §III-C is that it does not enable lower granularity of the preserved privacy—which is especially required when both the adversary and honest users us the same AP—unlike the protocol described in §III-B. To enable lower granularity in the protocol described in §III-B, we maintain several states in the router, which result high overhead that can be misused, whereas the protocol in §III-C reduces this overhead at the cost of reduced granularity. We propose a new algorithm in Algorithm 3 aiming to maintain the advantage of both protocols, by maintaining and distributing these states concerning access patterns of individual users at the APs, which usually are located closer to but not controlled by end users.

The main idea of the protocol is to distribute state $\varphi(u, n)$ on the AP associated with users generating such requests, and to store the face state $\varrho(f, n)$ in the router. Decisions for access privacy are made at the router with the help of the AP. When the AP receives a request from the user, it checks if the user requested the content before. If not, the pmode value is discarded (to eliminate possible cheating attack about pmode), and the AP forwards the request to the router. Otherwise, the AP directly sends the interest to the router. Upon receiving the interest from a given face, the router initially looks if the content is in the cache or not. If not, it retrieves the content from the origin server and serves it to the requesting user through that face; otherwise, the router checks the face state $\varrho(f, n)$: if it is zero, which implies that no user on that face has requested the content, the router returns the content after a delay $td(n)$ expires; otherwise, it looks at the flag generated by the AP: if it is true, which means that the user has already requested the content before, the router fulfills the interest immediately; otherwise, the interest is fulfilled after a delay $td(n)$ is expired.

Algorithm 3: Low granularity approach to preserving the privacy of cache access. The protocol uses the toy example in Figure 1.

```
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 interest Ints with pmode enabled for $n$. $u$ records $ts_0$
   and associate it with that request. $u$ sends the request to 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
```

IV. 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 perform several measurements on the CCNx prototype [26] using simulation setting. To derive an accurate representation of real-world timing scenarios, we feed the simulator with topologies and per-hop RTT traces driven from the current Internet. We do so using traceroute [27] to request several websites as shown in the details below.

A. Settings and Timing Data-sets

Our main measurements are based on CCNx, which is an open source system that implements the basic operations of CCN. CCNx implements both the communication operations—i.e., specifying naming and data conventions of communications exchanged between consumers and publishers—and security operations (using OpenSSL)–signature generation and verification for data packets.

Because no ICN architecture or design has been widely deployed yet, we lack any real-world traces of RTTs for content retrieval networks. However, designs like CCN suggest operating CCN on top of IP, making today's IP timings relevant for experiments. To this end, we instrument the CCNx simulator with real-world per-hop round trip delays when issuing interests from within our campus (connected directly to the Internet backbone) to reach each of the Alexa top-100 sites [2]. We use traceroute to obtain per-hop RTT delay to each of these sites—each of these sites is an origin server.

We notice that traceroute has several limitations that prevent direct use of its measurements in our study. First, as the hop count increases, there is no guarantee to have larger RTT than previous RTT for smaller hop count. Second, path to origin servers may change at any time, making two measurements for the same route greatly different. Last, traceroute provides cumulative RTT as the hop count increases, but not the per-hop time delay needed in our study. To address the first issue, we run many traceroute requests at different times of the day to account for different network conditions (which is the main reason that raises this issue), and record different readings for a fixed path to the requested site. Then, for each hop we consider the median RTT among all RTTs given for that hop. We observe that as we increase the number of measurements of the traceroute for the same site we get ordered set of (median) readings: the closer to destination the hop count is, the larger the RTT. To address the second issue of traceroute, we only consider the path that is most popular in the returned traceroute results, and discard all other paths. Once both issues are addressed, we compute the per-hop delay RTT_i as:

$$RTT_i = \begin{cases} RTT_i^t - RTT_{i-1}^t & i > 1 \\ RTT_1^t & i = 1 \end{cases},$$

where RTT_i^t is the i-th returned record by traceroute for the given site. A CDF of the per-hop RTT on the path to each of these origin servers is shown in Figure 3, with per-hop RTT values ranging from as low as parts of a millisecond to more than a hundred of milliseconds. Complementary CDF for smaller range of RTT ($RTT_i \leq 1$) is shown as a small graph within the CDF in Figure 3 (70% of the hops' RTT in all per-hop measurements are less 1 ms). Notice that the per-hop RTT are smaller than expected on the Internet, which might be due to that two hops are in the same router, same datacenter, or same CDN. However, the results in this study are less significantly affected by other than the total RTT (used for deriving $td(n)$) and the first hop delay (used for validating the attack). In the future, we will consider other less popular sites, where some of these issues can be eliminated, and request them from different geographical locations (e.g., using PlanetLab nodes) to diversify the measurements.

We feed the per-hop RTT to a dummy CCNx topology corresponding to the toy example in Figure 1 for each of the hop counts and the per-hop RTT to request these sites. That is, in each case we control the number of hops between router r_2 and r_1 in Figure 1 to correspond to the number of hops returned by traceroute for the given site. We then add the delay incurred over that hop as measured by traceroute using the method explained earlier.

Because the hop count to reach different sites varies from one site to another, we limit our attention to 24 sites that had exactly 16 returned valid hops in traceroute to unify our analysis and discussion in this section. We only limit our attention to those sites where traceroute returned 16 unmasked hops and discard timed-out hops, if any. A boxplot of the normalized per-hop RTT (defined as $RTT_i/\max\{RTT_k\}$ for $1 \leq k \leq h$ and h is the hop-count, where RTT_i is the i-th

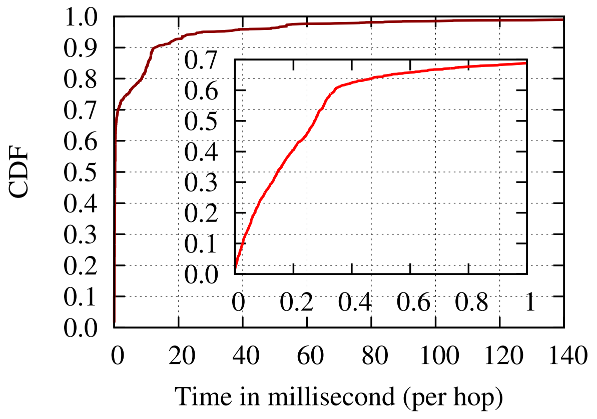

Fig. 3. An empirical CDF of the per-hop round trip delay for all hops of Alexa's top 100 sites using traceroute. Notice that 70% of the per-hop RTTs are less than 1 millisecond.

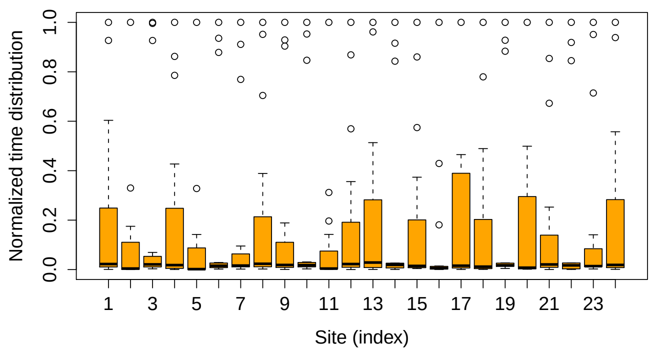

Fig. 4. Boxplot for the (normalized) distribution of per-hop RTT for each of the sites used in our study (indexed)—the boxplot plots the 1st, 2nd, and 3rd quartiles as well as the minimum, maximum, and outliers.

hop—notice that $RTT_1 = 2td_0$ in our protocols) for each of the 24 sites is shown in Figure 4. Finally, we define the RTT up to each hop as the sum of the per-hop RTT normalized by the total RTT to the origin server. This is, RTT_k is defined as

$$RTT_k^t = \sum_{i=1}^{k} RTT_i/RTT_h,$$

where $RTT_h = 2td_0 + td_x$ is the total RTT up to the origin server, and RTT_i is the the i-th hop RTT. Notice that RTT_k^t is returned by traceroute for each k, and can be used immediately in this study. A boxplot of the RTT up to each hop (1 to 16; until reaching the origin server) as a ratio of the total RTT to the origin server is shown in Figure 5.

B. Results

In this subsection we introduce the results of this study. We mainly verify how accurately an adversary can validate the attack and the overhead of our protocols in terms of added delay to the responses of benign users.

1) Attack validation: First, we examine whether an adversary co-located one-hop away from a legitimate user is 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 in CCN, contents are

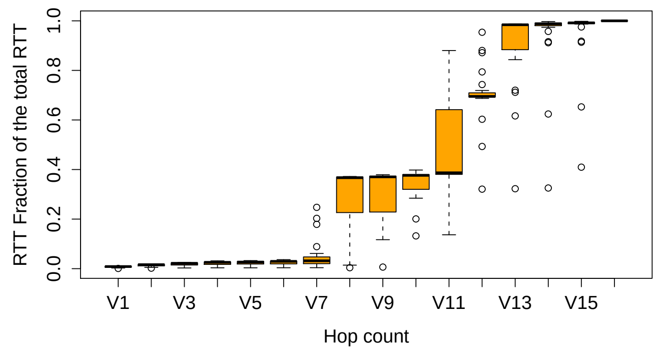

Fig. 5. A boxplot of the RTT (up to the given hop count) as a ratio of the total RTT (up to the origin server of the site) for 24 sites in Alexa's top 100, with 16 hops to each site.

replicated and cached at each hop, thus future requests are fulfilled immediately from the closest router to the user. From Figure 5, we observe that an adversary who is co-located with the user who has requested these sites 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 sites (and more than 99% for the median site). Even when a cache-miss happens, an RTT by a cache hit at the sixth hop away from the user, for example, would be 40 times at average (and about 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 user access patterns since a 6-hop network has a large anonymity set. By feeding the timing profiles in Figure 4 in CCNx we observe that the network latency is the dominating part of the RTT in CCN, and other ICN-related delay is negligible. From that, 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 due to their access be a potentially co-located user with him.

2) How defenses impact the performance: Now we look at how our protocols impact the performance of the ICN. One critical parameter for our designs is $td(n)$, which corresponds to the number of hops d that an edge router estimates and according to which he generates noise and uses it to fulfill pending interests issued by end users while maintaining privacy of prior requests. This parameter is generated and used in the three different protocols proposed in this work. Given that we have access to the per-hop delays (as shown in §IV-A), we use $d \leq h$ directly to compute $td(n)$ instead of the approximation in Eq. 1. To understand the impact of different values of d we define the maintained RTT gain metric as the difference between the gain in RTT due to caching for subsequent interest fulfillments (when contents are cached 1-hop away from the requesting host) and the the incurred delay due to the added noise in our protocols at a given d. This maintained gain is especially significant to benign users requesting the contents in the future. By observing that the first hop's RTT is negligible (as in Figure 5), we consider the maintained RTT gain (normalized) as $1 - (td(n)/td_x) \approx 1 - RTT^t_d$. We compute this quantity for the min, max, mean, and median RTT^t_d of the different sites, for different d values.

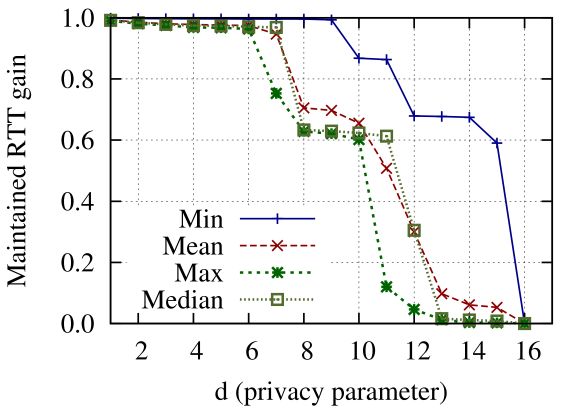

Fig. 6. Maintained RTT gain for different values of the privacy parameter d. Notice that the average (and more than 50% of sites, i.e., the median) of the maintained gain in RTT is 80% even when d is 50% of its maximum value.

Even when the router has the capability to record a per-hop RTT and add a given number of hops as noise—not an estimate as described in the protocols, the overhead as additional time delay added to the RTT of fulfilling requests to users still maintains the benefits of ICN as shown in Figure 6. For example, when $d = 6$ (which is one-third of the hop count to the origin server thus providing high anonymity set), a request to an average site would be fulfilled about 40 times faster than retrieving contents from the origin server (0.975 gain). Even for the site with the longest RTT, it would be 25 times (0.96 gain) faster than getting contents from the origin server. Even when d increases the results are not affected greatly: for $d = 7$, the mean, median, and max gain are 0.965, 0.97, 0.75, respectively. Similarly, for $d = 8$, we obtain 0.7, 0.633, and 0.62, respectively. However, as d reaches a value that makes the path traverse the core congested network with high TTL, this result degrades greatly: the performance worsen to reach an average gain of 0.5 at $d = 11$. As before, RTT is dominated by network latencies, whereas CCNx delays are negligible, supporting our claim that our designs maintain ICN's gain in RTT, and that the performance is tunable depending on the desirable privacy guarantees to provide to users.

3) How different network conditions affect the performance: Both of the previous sections make conclusions that are network-dependent. Accordingly, we perform similar requests from another commercial campus network that is separated from the Internet backbone by several hops, where several middle boxes are used for security purpose (the average total RTT has increased in these measurements by 300%). In these measurements we observe that the first hop would at average constitute 1% of the overall RTT, making the attack easily applicable, and the maintained gain for $d = 6$ in sites that have 16 returned hops by traceroute is 0.88 at average (8 times faster than retrieving contents from the origin server). We further make similar measurements by performing those requests from a residential network, and find a similar RTT for the first hop, although the gain for $d = 6$ for similar set of sites is about 0.92 at average.

4) Overhead evaluation: Evaluating the overhead at the routers and APs would depends greatly on how often contents are flagged as privacy related. Since we assume that a user who uses the pmode with requests is trusted, the overhead is a good estimate of real privacy needs. Misuses that try to exploit that and generate excessive overhead on routers can be penalized by feedbacks from other users. We notice that the last protocol, which outperforms all others, have limited overhead on routers. Also, we emphasize that there is no overhead on the network, since the delay generated would not affect the location of contents in the cache, but the time at which an interest is fulfilled.

V. Discussion

Our protocols make use of certain assumptions to enable the privacy of user access patterns in ICN. One of such assumptions is that users are willing to give up part of ICN (e.g., CCN and NDN) gains for their privacy improvement. With that in mind, and using our measurements showing that $d = 6$ in our protocols would still maintain more than 97% of the gains in CCN performance, our simulation shows and supports our protocol's usability. Particularly, we claim even $d < 6$ is large enough to provide a good anonymity set and to pronounce the timing attack ineffective.

Another assumption we make is that both the adversary and the benign user are residing behind the same router, and are 1-hop away from each other. On the other hand, if both users are 2-hops away, the adversary will still be able to infer some information about the co-location of the benign user who has requested the contents. We address this issue in two ways. First, given that the first few hops (as shown in Figure 5 have small RTTs, the adversary has to have a very sensitive measurements capability at the microsecond level to be able to tell if the user is 2, 3, or 4 hops away). Second, we believe that even in current networks which have many subscribers to the same routing infrastructure, 2-hop away users could likely be hidden in a large enough anonymity set. This makes it hard for the adversary to pinpoint a smaller set of users who could be potentially the requesters of the contents. Finally, although at the cost of additional overhead, one can extend our protocols to address this shortcoming.

An explicit assumption we make is that the adversary cannot collude with routers. However, two users acting as adversaries may collude with each other and try to bypass our defenses. For example, each of the colluding malicious users could issue an interest for a certain content, and compare their timings to infer whether the content has been cached (and that the router is generating noise as delay) or not. We notice that such collusion, while in principle is applicable to the first protocol, is not applicable to both the second and third protocol. As both requests have to go through the same face, they will both be considered as if they are from the same entity, regardless to the users who issued them (lines 13 and 14 in Algorithm 3 and lines 7 and 9 and 11 to 14 in Algorithm 2).

A closely related attack is what we coin as the "intersection attack", in which two geographically distributed attackers collude to infer if a content is cached or not. For example, suppose that one node that belongs to the attacker obtains a delay that tells the content is cached 3 hops away from that node. Another node that also belongs to the attacker which is 3 hops away tries to simultaneously requests the same content, and obtains the same delay, from which both colluding nodes will know that the content cannot be cached three hops away from each of them at the same time. However, in order for this attack to work, the attackers need to: 1) be geographically distributed, and 2) know in advance the path on which the interests of benign users have reached origin servers. While the first requirement would violate one of our attacker model assumptions, we believe that the second requirement would require collusion of the underlying infrastructure (routers) or much larger number of attackers to make a good estimate of the path. Even though the attack is possible in theory, our defenses and privacy protection mechanisms raise the bar greatly for such adversaries in practice.

VI. Related Work

There have been several directions of work in the literature related to the work we present in this paper. These work are information centric networking architectures, caching mechanisms, and security evaluation and analysis of these mechanisms and architectures. To the best of our knowledge, our work is the first to address the timing attack by exploiting caching mechanisms in ICN.

Other architectures that date prior to the introduction of CCN [17] and NDN [33] include TRIAD [15] (the work that pioneered the concept of ICN), ROFL [8], and DONA [21]. Other recent architectures include XIA [3], [16], CONET [11], DACON [20], and SCION [35] (where the latter mainly addresses routing). Some of these architectures do not specify how caching is implemented whereas others do. For the latter type of architectures, we will look forward to extend our work in the future by examining if they are vulnerable to the attack introduced in this paper. For details, we direct the reader to a survey on some of these architectures and a comparison between them in [1].

Caching has enjoyed a cornerstone position in the ICN research community, since it is one of the main features that ICN advocate as an advantage over the current Internet. The main motivation of such caching algorithms introduced in the literature is performance, rather than privacy. Most of schemes introduced in the literature for caching aim to reduce the RTT and improve users experience by maximization of cache hit and minimization of cache-miss. Prior literature on caching in ICN include the work in [9], [10], [19], [23], [25], [29], [31]

A critique of caching mechanisms, as suggested in CCN and NDN—as well as other issues worth investigation to enable ICN architectures—is introduced in [13]

Security and privacy of ICN have been discussed in several recent works. In [32], secure naming system has been proposed. Named-based trust and security protection mechanisms are introduced in [34]. Different naming conventions in ICN architectures and their security features are discussed in [12].

A privacy-preserving contents retrieval in ICN (that assumes the origin server is dishonest) is proposed in [6]. A diverse array of security mechanisms for ICN is introduced in [18]. A closely related architecture that makes accountability as a first-order property, named AIP, is introduced in [4] (which shares similarities with the naming in [7]. Arguments for ICN and future Internet design in general are in [14], [28]

VII. Concluding Remarks

In this paper we have introduced an attack on content access privacy that is applicable to several information centric network (ICN) architectures, including the CCN and NDN. We show that an adversary with the capability to perform timing measurements can infer whether contents have been fetched by other users by exploiting the universal caching mechanism deployed in such architecture. We verify such attack theoretically and empirically using real-world per-hop time measurements.

To withstand such attack, we introduce three protocols, each of which comes at varying cost and benefits to the network. In these protocols, we make use of carefully chosen time delay to responses given by routers to fulfill requests by users. The delay is chosen to strike a balance between the amount of privacy provided to users—which is determined by the delay added to increase a number of virtual hops away from the user requesting privacy-related contents, the overhead on routers, and the degradation of service to benign users.

One limiting factor of our attack is the cache update pattern in real-world edge routers, for which we do not have any real-world measurements. In the future, we will look at how different caching policies and cache flushing patterns (and the time associated with that) would affect the effectiveness of both the attack and the defenses we provide in this work. In another future work, we will look at how other caching algorithms [9], [10], [19], [31], which are tailored specifically to improve the cache hit in ICN architectures, are prone to the attack we proposed in this work, and will look for defenses to mitigate it, if applicable.

References

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Comm Mag.*, 2012.

[2] Alexa. Top 500 sites. http://www.alexa.com/topsites, July 2012.

[3] A. Anand, F. Dogar, D. Han, B. Li, H. Lim, M. Machado, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste. Xia: an architecture for an evolvable and trustworthy internet. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 2:1–2:6, New York, NY, USA, 2011. ACM.

[4] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable internet protocol (AIP). In *Proceedings of the ACM SIGCOMM*, 2008.

[5] T. Anderson, K. Birman, R. Broberg, M. Caesar, D. Comer, C. Cotton, M. Freedman, A. Haeberlen, Z. Ives, et al. Nebula-a future internet that supports trustworthy cloud computing. *White Paper*, 2010.

[6] S. Arianfar, T. Koponen, B. Raghavan, and S. Shenker. On preserving privacy in content-oriented networks. In *Proceedings of the ACM SIGCOMM ICN*, pages 19–24, 2011.

[7] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A layered naming architecture for the internet. In *Proceedings of ACM SIGCOMM*, pages 343–352, 2004.

[8] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica. Rofl: routing on flat labels. In *Proc. of ACM SIGCOMM*, 2006.

[9] W. Chai, D. He, I. Psaras, and G. Pavlou. Cache "less for more" in information-centric networks. *NETWORKING 2012*, pages 27–40, 2012.

[10] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, and S. Pack. Wave: Popularity-based and collaborative in-network caching for content-oriented networks. In *Proceedings of IEEE INFOCOM Workshops*, pages 316–321, 2012.

[11] A. Detti, N. Blefari Melazzi, S. Salsano, and M. Pomposini. Conet: a content centric inter-networking architecture. In *Proceedings of ACM SIGCOMM ICN*, ICN '11, New York, NY, USA, 2011. ACM.

[12] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in content-oriented architectures. In *Proceedings of ACM SIGCOMM ICN*, pages 1–6, 2011.

[13] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of ACM HotNets*, 2011.

[14] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Intelligent design enables architectural evolution. In *Proceedings of ACM HotNets*, 2011.

[15] M. Gritter and D. R. Cheriton. An architecture for content routing support in the internet. In *USITS*, pages 37–48. USENIX, 2001.

[16] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste. Xia: efficient support for evolvable internetworking. In *Proceedings of USENIX NSDI*, 2012.

[17] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. Braynard. Networking named content. In *Proceedings of ACM CoNEXT*, pages 1–12, 2009.

[18] J. Jeong, T. T. Kwon, and Y. Choi. Host-oblivious security for content-based networks. In *Proceedings of ACM CFI*, pages 35–40, 2010.

[19] K. Katsaros, G. Xylomenos, and G. Polyzos. A hybrid overlay multicast and caching scheme for information-centric networking. In *Proceedings of IEEE INFOCOM*, 2010.

[20] D. Ko, K. Cho, M. Lee, H. Kim, T. T. Kwon, and Y. Choi. Decentralized and autonomous content overlay networking (dacon) with wifi access points. In *Proceedings of ACM CFI*, pages 18–24, 2010.

[21] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM CCR*, 37(4), 2007.

[22] 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 *Proceedings of ACM CCS*, pages 199–212, 2009.

[23] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi. Transport-layer issues in information centric networks. In *ACM SIGCOMM ICN*, 2012.

[24] I. Seskar, K. Nagaraja, S. Nelson, and D. Raychaudhuri. Mobilityfirst future internet architecture project. In *Proc. of ACM AINTEC*, 2011.

[25] S. Singh. A trust based approach for secure access control in information centric network. *Int'l J. of Info and Network Security*, 1(2), 2012.

[26] The CCNx Project. CCNx. https://www.ccnx.org/, July 2012.

[27] Traceroute. Traceroute. http://www.traceroute.org/, July 2012.

[28] D. Trossen, M. Sarela, and K. Sollins. Arguments for an information-centric internetworking architecture. *ACM CCR*, 40(2), 2010.

[29] G. Tyson, N. Sastry, I. Rimac, R. Cuevas, and A. Mauthe. A survey of mobility in information-centric networks: challenges and research directions. In *Proceedings of ACM MobiHoc Workshops*, 2012.

[30] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp. Backscatter from the data plane — threats to stability and security in information-centric networking. *CoRR*, abs/1205.4778, 2012.

[31] Y. Wang, K. Lee, B. Venkataraman, R. Shamanna, I. Rhee, and S. Yang. Advertising cached contents in the control plane: Necessity and feasibility. In *Proceedings of IEEE INFOCOM Workshops*, 2012.

[32] W. Wong and P. Nikander. Secure naming in information-centric networks. In *Proceedings of ACM ReARCH*, pages 12:1–12:6, 2010.

[33] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton, D. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. Technical report, PARC, 2010.

[34] X. Zhang, K. Chang, H. Xiong, Y. Wen, G. Shi, and G. Wang. Towards name-based trust and security for content-centric network. In *Proceedings of the IEEE ICNP*, pages 1–6, 2011.

[35] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen. Scion: Scalability, control, and isolation on next-generation networks. In *Proc. of IEEE S&P*, pages 212–227, 2011.