Timing Attacks on Access Privacy in Information Centric Networks and Countermeasures

Aziz Mohaisen, *Member, IEEE,* Hesham Mekky, *Student Member, IEEE,* Xinwen Zhang, *Member, IEEE,* Haiyong Xie, *Member, IEEE,* and Yongdae Kim, *Member, IEEE*

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 cost and benefits, and raise the bar for an adversary to apply such attack.

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

1 Introduction

Information centric networks (ICNs) are new Internet architectures to secure and efficiently disseminate contents. In several ICNs, such as content centric network (CCN) [24] and named data network (NDN) [45], contents are fetched by their names from caches in the network or from origin servers. Once the content is fetched from an origin server, it is replicated and cached in all routers along the routing and forwarding path, starting from the user to the origin server, thus allowing further interests with the same content name to be fulfilled quickly [24]. For example, when another user issues an interest in contents that have been previously served to a user on the same path, the interest is fulfilled from a near-by cache. This design choice is an essential building block of ICNs for reducing latency [24], [45].

The universal caching mechanism in ICNs poses a privacy risk [3], [29], [31]. In particular, the time difference between responses of cached and uncached content can be used as a side channel to infer whether a near-by user has previously requested the same content or not.

- *A preliminary version of this work appears at ACM ASIACCS'13 [32] and in the poster session of ACM CCS'12 [31].*
- *A. Mohaisen is with Verisign Labs, Reston, VA 20190, USA. Email: amohaisen@verisign.com*
- *X. Zhang is with Samsung Telecommunications America, CA, USA.*
- *H. Xie is with Huawei Technologies, CA, USA and UTSC, China.*
- *Y. Kim is with the Electrical Engineering Department, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea.*

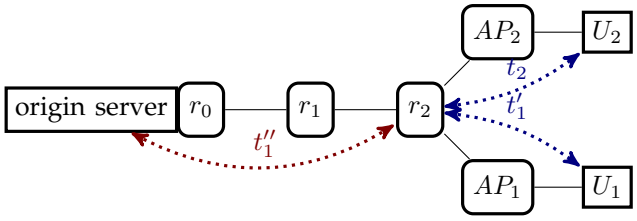


Fig. 1. Toy example of timing attack in ICN. $t_1 = t_1' + t_1''$. A user U_2 can infer whether user U_1 has accessed a content based on the different in RTT for cached and uncached content. AP_1 and AP_2 are access points

1.1 An Example of Attack on Privacy in ICN

Consider the topology in Figure 1, where U_1 and U_2 are users connected via routers r_0, r_1 and r_2 (each with its own cache) to an origin server holding a content n. Suppose that U_2 is an adversary while U_1 is an honest user. If U_1 issues an interest n that resides behind r_0, the interest traverses the path $U_1 \rightarrow AP_1 \rightarrow r_2 \rightarrow r_1 \rightarrow r_0$. According to the routing properties of ICN [24], the response to the internet is sent over the returning path $r_0 \rightarrow r_1 \rightarrow r_2 \rightarrow AP_1 \rightarrow U_1$. 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, when U_2 request n, the path that the interest would traverse is $U_2 \rightarrow AP_2 \rightarrow r_2$, and the contents would return on the reversed path of 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 might seem difficult, today's Internet topology shows the applicability of the attack (see §4.2.4), if ICN is to be built on top of today's IP networks (a direction suggested in [15]). Furthermore, pinpointing individual users might not be necessary in many advanced applications: an adversary in a business intelligence attack might be more interested in knowing what contents are 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, while using the above technique.

1.2 Shortcomings of Simple & Existing Solutions

The caching problem discussed earlier is not limited to ICN, but is also seen in other systems, such as web services. However, the attack we describe in this paper is off-path, whereas the attack most common in the web privacy literature [16] is on-path: an adversary has to proactively participate in servicing contents to the potential victim in order to know if she already accessed contents of a third party (utilizing the timing side channel) or not. Second, web caching is not universally deployed as a feature of today's web, and private browsing mode in most of today's browsers enables users to disable caching for privacy, whereas disabling caching [16] is against the basic design principles of ICN. Finally, the attack in [16] is only applicable on legacy browsers and web architectures, whereas today's javascript enforces same-origin policy and Java (6 and above) plug-ins utilize separate cache from the web browser's cache.

Efficiency-oriented selective caching [12] is limited in practice: if a router wants to decide caching based on how far a user is away from it, it has to know the user's location in advance or have the location provided to the router at the time of caching. The first approach is unrealistic and requires a global knowledge of the network, while the second approach is vulnerable to misuse. Similarly, popularity based approaches [13], [26] are also vulnerable to misuse. For example, 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; cf. §5.3.

Concurrent to this work, the same problem was pointed out in [3], [29], and the solutions provided to the problem evolve around making names unpredictable, using systems like Tor, or adding delay equal to the delay when serving the contents from the origin server. The authors of [3] particularly suggest adding the time noise intelligently: a cache-miss is triggered uniformly or exponentially random with respect to the requests. Making names unpredictable is non-trivial and undesirable in ICN [18]. A tunneling system would require Tor to be scalable to bootstrap the Internet, another unfounded assumption [30]. A simple solution that adds the same amount of time delay, as admitted by the authors in [29] is against the idea of ICN. A solution that generates cache-miss randomly, as admitted by the authors of [3], is only probabilistic in what it provides the user of privacy guarantees, and some users (with consecutive cache-hit) are provided no privacy.

Modifying the policy not to cache flagged contents for privacy [29] is vulnerable. For example, 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 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.

1.3 Contributions and Organization

We examine timing attacks on ICN caching and propose three solutions that come at varying costs and complexities. Our main approach relies on randomly generated time paddings to disguise responses for interests issued by users in the same domain, thus increasing the anonymity set, defined as the set of domains and users that such requests potentially come from. While a similar attack is pointed out earlier in [16] for web caching, and timing is applicable as a side channel in anonymous communication systems [23], we claim the novelty of the attack on ICN, as in the two other concurrent works in [29] and [3]. Furthermore, we claim the novelty of the solutions provided in this paper, with their specific details to the problem at hand.

- We demonstrate timing attacks on the universal caching mechanism in ICN designs using CCNx, a prototype implementation of the CCN [38].
- We propose three protocols, each with different levels of complexity and privacy guarantees that prevent an adversary co-located with the benign users to infer whether they have accessed certain contents or not by relying on timing differences.
- We provide an evaluation of the fundamental premises of the solutions and attack, including an analysis of networks where the attack is applicable and the guarantees are realizable.

Organization. In Section 2, we review the preliminaries and terminologies. In Section 3, we introduce three protocols to address the problem and maintain the privacy of users access patterns. In Section 4, we present our simulation results to validate the attack and evaluate the performance of our proposed protocols. In Section 5, we highlight several discussion points, including potential attacks and their applicability to our protocols. Section 6 reviews related work, and Section 7 concludes this work and points out our future work.

2 Preliminaries and Terminology

In this section, we first review the terminologies related to ICN architectures—with CCN and NDN in mind—in §2.1. We then review the attack model used in this paper in §2.2. We informally describe what we mean by privacy in §2.3, and finally list our design goals in §2.4.

2.1 Terminologies and ICN Operations

In ICN, contents are fetched by their names [24]. 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. Each interest has a unique interest identifier, PID. 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.

The main features we utilize in our work from the design of ICN are universal caching and back-traceable routing. While the first feature enables that attack, back-traceability allows us to reason about quantified benefits of our protocols: the number of hops up and down from the user to the origin server is maintained.

2.2 Attack Model

We consider an adversary co-located with an honest user trying to access contents from ICN. We emphasize that the attack is focused and targeted, where there will always be some users within close proximity of the adversary who are possibly of interest to his inference attack. We assume that the adversary can perform fine-grained time measurements. In our model, names are predictable, and the attacker has a list of potential "names" that could be accessed by a user. We exclude insider attacks, or major (state-sponsored) adversaries, thus it follows that the adversary has no control over the path an interest traverses. The attacker cannot be geographically distributed to perform intersection attacks (cf. 5). We assume that the caching is utilized, which allows the adversary enough time to perform the attack.

Identities are random, and it is computationally hard to guess them. A user residing in a different domain who wants to infer the behavior of another user cannot perform an identity cloning. The operation of some of our protocols rely on the use of a weak form of identity that is randomly generated for the purpose of communicating privacy-related contents. Thus, an attacker cannot eavesdrop on communication within a different network, nor launch a replay attack: an adversary interested in performing replay attacks needs to access timing information and the interest-specific identity.

Attack scope. The attack is applicable to CCN, and to a lesser extent to other architectures [8], [22], [36]. We note several efforts for improving caching in ICN. Although, they are motivated by performance only; they improve user experience 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 shown in [42]. Verifying how vulnerable are these architectures to our attack is left as a future work.

2.3 Privacy Definition and Evaluation Criteria

We use the classical definition and meaning of the privacy as "anonymity": the adversary should not be able with reasonable resources to pinpoint the user fetching such contents among a finite number of users within his proximity. To this end, we definite the anonymity set of a user fetching the contents as the number of users at a less than or equal distance from the adversary to that user—who could potentially be fetching that same contents. The total ideal anonymity set size is the total number of users in the ICN, although we relate the set to other limited scopes, like country and city (cf. §4.2.4).

The privacy notion highlighted above can be stated and realized as follows. For an adversary to make use of the timing information exposed by the caching mechanism proposed by ICN paradigm, he would be interested in knowing the maximal probability that a content, named $\mathcal{N}$, is fetched by user u_i, given that the timing observed by the adversary $\mathcal{A}$ is t_0 to fetch that same content. That is, the adversary would like to compute the probability $P_r(\mathcal{U} = u_i | t = t_0, \mathcal{N})$. For simplicity, we assume a single name of interest, and so the adversary's probability is reduced into $P_r(\mathcal{U} = u_i | t = t_0)$. For example, if the adversary and the user reside in the same /24 network, and the measurements obtained by the adversary indicate that she resides within the same network, then the user has an ideal anonymity set of 254 (total number of usable addresses in the network), thus $P_r = 1/254$. Further quantification of privacy is in §4.2.4.

2.4 Design Goals

The goal is to protect the user privacy in ICN at a reasonable cost, utilizing the following explicit goals:

- Privacy protection: preventing an adversary from inferring other users' access patterns to contents fetched and distributed using ICN (details below). The privacy is defined in § 2.3.
- Cost effectiveness: the overhead (computation, communication, and memory) used for privacy protection should be reasonable in relation with the operational overhead of ICN operation.
- Minimal change to existing ICN protocols: ideally, we want our solutions not to alter the way caching and routing operate in ICN. We do respect that by maintaining as much as possible of the universal caching feature which enables performance in ICN.

TABLE 1
Fine Grained Privacy Approach Summary

	1st Request	Private	Flagged	Delay
Cached	✓	✓	✓	✓
	✗	✓	✗	✗
Not Cached	—	—	—	ICN

3 Protection Mechanisms

The first protocol, 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 (from a *normal distribution* with proactively learned parameters) to simulate a network delay before sending 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 3.1.

To reduce the overhead in the vanilla protocol a coarse-grained protocol is proposed by allowing routers to maintain per-face states instead of per-user states. In this efficient approach, an edge router generates random delay and serves the content upon its expiration when an interest of a cached content arrives for the first time at a certain face. 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 fine-grained privacy preservation. The detailed protocol is in §3.2.

To enable fine-grained privacy protection, while reducing 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. When a request is issued by a user behind an AP, the access point maintains his identifier and the number of times she previously requested that content. If it is the first time request, the content is previously marked for private query, and the request of that content is flagged, the router generates random delay from a normal distribution then serves the content to the user via the AP. If the content is previously marked as private and the user has already requested the content before, the request is not flagged, then the contents are served directly to the user. If the content is not cached, and regardless to the flag state, it is served according to the original ICN protocol. This process is summarized in Table 1. This approach maintains the privacy of the requester from the same domain, while reducing the states stored on the router for face statistics, whereas all user statistics are stored on the close by AP. The detailed protocol is in §3.3.

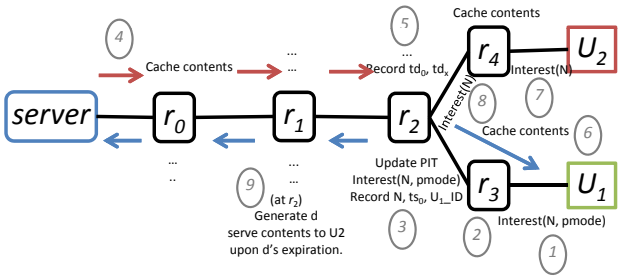


Fig. 2. An illustration of the protocols in Algorithms 1 through 3. State updates as well as repeated steps (at r_1 and r_0) are omitted for brevity.

Delay generation. The procedure is performed by an edge router. The procedure takes several parameters for generating td, the time delay that in theory determines the number of hops, d, where the contents are cached away from the user. Let $n \in N$ be a content name, h be the total number of hops to the origin server, td_x be the RTT to the origin server, td_0 be the first-hop time delay. We define $td(n)$ for all subsequent requests as:

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

Note that $td(n)$ is generated only once from a normal distribution, similar to the actual RTT distribution as shown in §4. Also note that d is only used in our analysis of the protocols, and the router does not have to know that value except for assurances of the privacy guarantees as used in our evaluation. For that, it is only sufficient for the router to be aware of td_0, which the router can estimate easily, and the a rough estimate of td_x, which is often known for various regions [1].

Finally, we note that all timing measurements are done by the edge router, the router that connects a user to the rest of the ICN. Thus, no timing distribution is needed, and time delay is added only once.

3.1 The "Vanilla" Approach

The vanilla algorithm to prevent timing attacks on privacy in ICN is described in Algorithm 1 and illustrated in Fig. 2. The main ingredient of the algorithm is a carefully chosen 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. For that, the protocol relies on states stored by each edge router to name the contents, the number of times the contents are served to each user, and the user ID.

For a user u (U_1 in Fig. 1), her edge router (r_2 in Fig. 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

Algorithm 1: The "vanilla" approach to preserving the privacy of cache access.

Input: n - a content name, u - a user, φ - access state, $Ints = (u, n, \mathsf{pmode}, ts_0)$
Output: Contents to u while preserving privacy.

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.
if $\mathsf{pmode} == 0$ **then**
  if $td(n) == 0$ **then**
    // default value $td(n) = 0$
    R follows ICN protocol to obtain data packet $Data$ from the origin server;
    R returns $Data$ to u;
  else
    R follows ICN protocol to obtain data packet $Data$;
    R delays $td(n)$;
    R returns $Data$ to u;
  end
else
  if $\varphi(u, n) == 0$ **then**
    if n *is cached* **then**
      R delays $td(n)$;
      R returns $Data$ to u;
    else
      R follows the ICN protocol to obtain data packet $Data$ from the origin server;
      R records ts_2 upon the arrival of $Data$, and computes:
        $td_x = ts_2 - ts_1$; // RTT from R to origin server
        $h = td_x/(2td_0) + 1$; // expected # of hops from u to the origin server
        Generate $td(n)$ according to Eq. 1;
        $\varphi(u, n) + +$;
      R returns retrieved $Data$ to u;
    end
  else
    R returns cached $Data$ to u;
  end
end

Algorithm 2: The efficient approach to preserving the privacy of cache access.

Input: n - content name, f - face id, ϱ - access state, $Ints = (n, \mathsf{pmode}, ts_0)$
Output: Contents to f while preserving privacy.

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.
if n *is not in R's cache* **then**
  R follows the ICN protocol to obtain data packet $Data$ from the origin server;
  R records ts_2 upon the arrival of $Data$, and computes:
    $td_x = ts_2 - ts_1$; // RTT from R to origin server
    $h = td_x/(2td_0) + 1$; // expected # of hops from f to the origin server
    Generate $td(n)$ according to Eq. 1;
    $\varrho(f, n) + +$;
  R returns $Data$ to AP via f;
else
  if $\varrho(f, n) == 0$ **then**
    R generates $td(n)$ as in Eq. 1;
    R delays $td(n)$.
  end
  R returns $Data$ to the AP via f;
end

$Ints = (U_1, n, \mathsf{pmode}, ts_0)$; pmode is a flag for privacy, with pmode = 1, where ts_0 is the timestamp of when the interest is issued. When r_2 receives this, it follows the ICN protocol [24] 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.

3.2 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 considered, where each domain may have large number of users—overhead evaluation is in §5.6. In order to reduce the overhead, 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 the same (sub-)domain would traverse the same face. Accordingly, per-face states are stored and maintained in each router, and decisions to preserve privacy are made based those states.

Algorithm 2 shows the protocol for an edge router. Unlike Algorithm 1, each router stores $\varrho : F \times N \rightarrow INT$, where F is the set of faces. The parameter $\varrho(f, n)$ indicates the number of times that content name n is requested from face f. The protocol can be illustrated on Fig. 2, where router r_2 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 potentially thwart the privacy of other users' access.

3.3 Fine-Grained Approach

The shortcoming of the previous protocol is that it does not enable fine-grained privacy, which is especially required when both the adversary and honest users use the same AP, and unlike the protocol in §3.1. To enable fine-grained privacy in §3.1, we maintain several states in the router, which result high overhead that can be misused, whereas the protocol in §3.2 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 distributing the states concerning access patterns of users with 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: Fine-grained approach to preserving the privacy of cache access.

Input: n - content name, f - face id, u - user id, ϱ - access state,
$Ints = (n, \mathsf{pmode}, ts_0, flag = false)$
Output: Contents to u while preserving privacy.

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.
When the AP receives *Ints*:
if $\varphi(u, n) == 0$ **then**
    AP discards the pmode tag and flags *Ints* with $flag = true$;
    AP forwards *Ints* to router R;
else
    AP forwards *Ints* to router R;
end
Upon receiving *Ints* from face f, the router R:
if *n is not in R's cache* **then**
    R follows the ICN protocol to retrieve the contents from the origin server and serve them to u.
else
    if $\varrho(f, n) == 0$ **then**
        R generates $td(n)$ with Eq. 1;
        R delays $td(n)$;
        R returns $Data$ to face f;
    else
        if $flag == true$ **then**
            R generates $td(n)$ with Eq. 1;
            R delays $td(n)$;
            R returns $Data$ to face f;
        else
            R fulfills the interest from cache
        end
    end
end

4 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 [38] 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 [39] to request several websites as shown in the details below.

4.1 Settings and Timing Data-sets

Our measurements are based on CCNx, an open source system that implements the basic operations of CCN. CCNx implements both the communication operations and security operations. Because no ICN architecture or design is 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 [5]. We use traceroute to obtain per-hop RTT delay to each of these sites, assuming that each is an origin server. We follow best practices to eliminate measurements of traceroute, including multiple runs at different times of the day to account for network congestion. Using the measures of

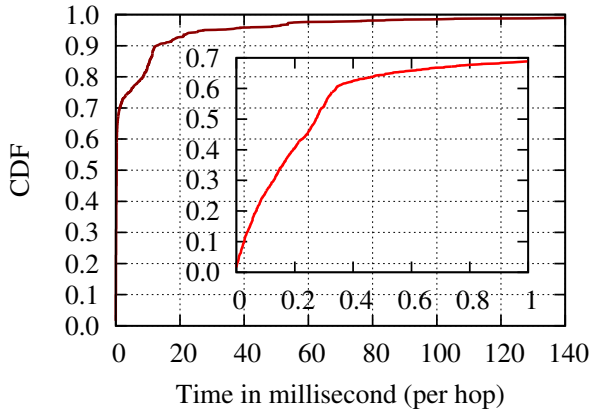


Fig. 3. An empirical CDF of the per-hop round trip delay for all hops of Alexa's top 100 sites, where 70% of the per-hop RTTs are less than 1 millisecond.

traceroute, 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 hop timing to that site. A CDF of the per-hop RTTs is shown in Fig. 3. Notice that the per-hop RTT is smaller than expected on the Internet, perhaps for that multiple 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).

We fed the per-hop RTT to a CCNx topology similar to the example in Figure 1 for each site. That is, in each case we control the number of hops between router r_2 and r_1 in Fig. 1 to correspond to the number of hops with their delays obtained in the previous measurements. Note that this is the most accurately representative scenario to CCN, because of its back-traceable routing.

Because the hop count to different sites varies, we considered 24 sites that had exactly 16 returned valid hops with timing information. A boxplot of the normalized per-hop RTT (defined as $RTT_i / \max\{RTT_k\}$ for $1 \leq k \leq h$) for each of the 24 sites is shown in Fig. 4. Finally, we define the RTT up to each hop as $RTT_k^t = \sum_{i=1}^{k} RTT_i / RTT_h$, where $RTT_h = 2td_0 + td_x$. Notice that RTT_k^t is returned by traceroute for each k, and can be used immediately in this analysis. A boxplot of the RTT up to each hop as a ratio of the total RTT to the origin server is shown in Fig. 5.

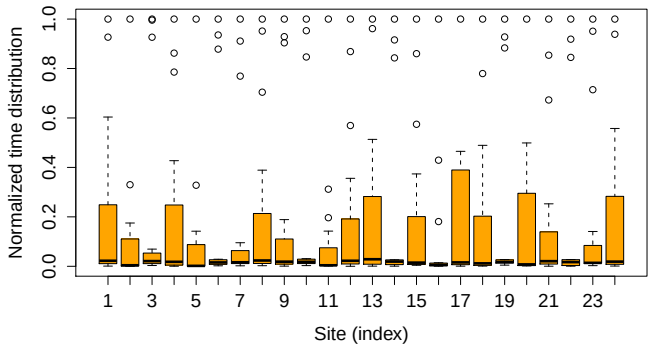


Fig. 4. Boxplot for the (normalized) distribution of per-hop RTT for each of the sites used in our study (indexed).

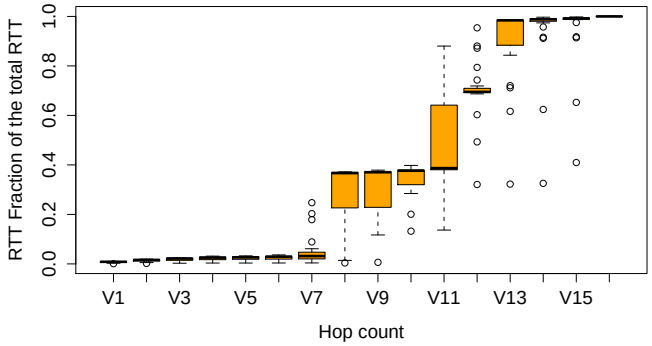


Fig. 5. A boxplot of the RTT (up to the given hop count) as a ratio of the total RTT (up to an origin server) for 24 sites in Alexa's top 100, with 16 hops to each site.

4.2 Results

4.2.1 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 replicated and cached at each hop, thus future requests are fulfilled immediately from the closest router to the user. From Fig. 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 (cf. 4.2.4).

By instrumenting 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 by a potentially co-located user with him.

Attack sensitivity: To test the sensitivity of the attack to network latencies, we ran the following experiment. We

selected three web sites for our measurements, namely Facebook's, Google's, and Verisign's. We measured the first and second hop timing for each site, repeated the experiment 200 times, and plotted the probability density functions in Fig. 6 for the resulting timing profiles. We notice that an adversary interested in characterizing where contents are stored will be able to easily identify the cache by relying on the time measurements; in all measurements there is a clear distinguishing gap between the timing results for the first and second hop. We further notice that this scenario of measurement represents an upper bound on the attack difficulty, since an adversary is typically separated by multiple hops from an origin server allowing a larger gap in the measured times.

4.2.2 How defenses impact the performance

One critical parameter for our designs is $td(n)$, the added time delay to maintain users' privacy. The value of $td(n)$ determines d, the number of hops virtually added by the router, which determines the new anonymity set for a user requesting the contents. $td(n)$ is generated and used in the three different protocols proposed in this work. For analysis purpose only, we use d as an indicator for anonymity (cf. §4.2.4). Given that we have access to the per-hop delays (cf. §4.1), we compute $td(n)$ that corresponds to the given $d \leq h$. As such, we enumerate all the possible values of $td(n)$ in Eq. 1. To understand the relationship between d and performance, we study the maintained RTT gain (defined as time effect of caching for subsequent interest fulfillments) for various d values. 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 Fig. 5), we define the (normalized) maintained RTT gain as $1-(td(n)/td_x) \approx 1-RTT_d^t$. We compute the min, max, mean, and median RTT_d^t of the different sites.

The results are shown in Fig. 7. We notice that fulfilling requests to users while maintaining privacy (at various d values) still maintains the benefits of ICN. For example, when $d = 6$, 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 significantly affected: 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.

4.2.3 How 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 on average constitute 1% of the overall RTT, making the attack easily applicable. Furthermore, we observed that the maintained gain for $d = 6$ in sites that have 16 returned hops by traceroute is 0.88 on average, corresponding to 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 a similar set of sites is about 0.92 on average. Notice that *we do not make any assumption on the structure of the RTT for the advantage of our protocols*. In particular, these mitigations will work with certain advantages (as opposed to not caching contents) regardless of the number of hops (as long as $h > 2$), and regardless to the per-hop RTT. They will, however, provide better performance when the RTTs are within a certain distribution, which happens to be the case for the measurements we performed. Similarly, the sensitivity of the adversary to detecting the attack relies less on the per-hop delay as an assumption, but the fact that the last hop towards the user constitutes a small fraction of the entire RTT to the origin server.

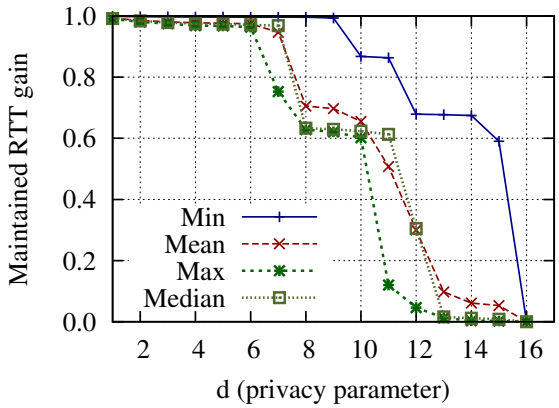
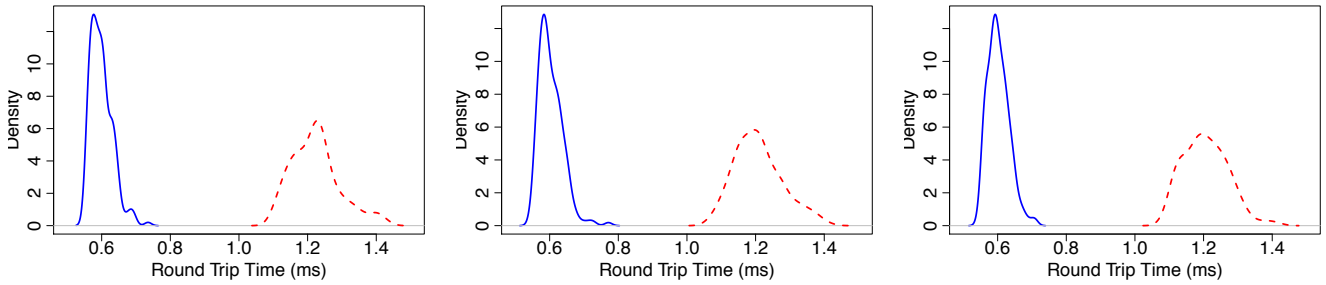


Fig. 7. Maintained RTT gain for different values of the privacy parameter d. Notice that the average (and more than 50% of sites, i.e., the median) of the maintained gain in RTT is 80% even when d is 50% of its maximum value.

4.2.4 Today's topology validates the attack

So far, we discussed the findings in this paper based on an ideal topology. Unfortunately, there is no information centric network in deployment to realize an operational scenario to understand the problem in practice. Furthermore, given that we do not have access to realistic numbers of users in a network to establish bounds on anonymity and privacy breach, we consider an analogous scenario that provides a bound using allocated resources (e.g., number of IP addresses associated with



(a) Facebook. (b) Google. (c) Verisign.

Fig. 6. Attack sensitivity. The solid curve is the probability density function (of 200 runs) for RTT up to the first hop whereas the dashed curve is up to the second hop.

a network). While one can use today's Internet topology and address allocations as bases for ICN deployment [15], it is unclear how often this ideal topology would exist in reality. The significance of the attack is only realized when it is applicable at scale. In order for that to happen in today's Internet settings, one of the following two conditions have to be satisfied:

- *Breaching individual users' privacy:* The number of addresses associated with a network (operated by an operator) are small enough so that the per-hop timing information can be used to infer a user behavior. Formally, this limited number of addresses translates into a small anonymity set that an adversary co-located with the benign request originator would exploit to breach the requester's privacy. The attack would not be possible when the number of addresses associated with an entity is large.
- *Business intelligence attack:* Multiple entities of a relatively small size (e.g., networks) share the same medium (e.g., wire and router) to access the ICN. For example, in the business intelligence scenario, the attack would only be meaningful if somewhat relatively small number of organizations share the same router to gain access to the ICN.

We note that both scenarios above use networks as an entity run by different operators for either identifying users within the same network (user privacy), or identifying users in different networks but using the same exit router to the Internet (business intelligence). The closest analogous scenario to the settings at hand is today's autonomous systems topology and address allocation associated with such systems. In both scenarios, we want to examine that there are networks with limited number of users, and other networks (perhaps small in number and size) that share the same exit router to the internet.

To understand the first scenario and its relevance on today's Internet, we map the whole space of IPv4 addresses to autonomous systems (ASes). Utilizing a geo-mapping dataset, we aimed to find the potential number of identifiable users by addresses within each network (ASN). To reduce the dimensionally of the problem, we limit our attention to /24 address mapping. In total, we found 8,527,812 /24 addresses covered in our dataset (about half of the IP space). We were able to map those addresses to 41,912 autonomous systems.

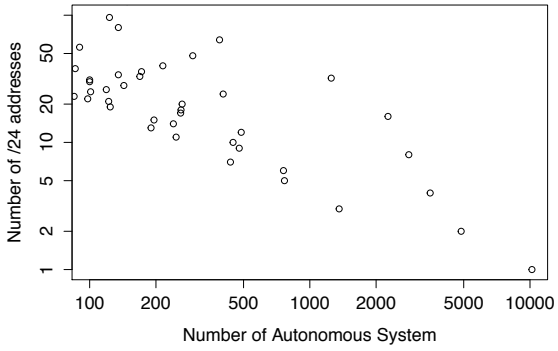


Fig. 8. Size of autonomous systems on the Internet.

Fig. 8 shows the tail of the distribution of the number of /24 addresses associated with ASNs. The x-axis shows the number of ASNs that have the /24 addresses shown in number on the corresponding y-axis (we trim the head of the distribution to highlight the problem). We note that almost 25% (10,207) of the ASNs have only one /24 address, which means a user using an IP address within the network has only an anonymity set of 254 (usable address in /24). In comparison, the same user would have an anonymity set of 1,512,160,298 (from 5,953,387 /24 addresses) if the same user resides in the United States and the whole country is covered (by adding delay of less than 45 milliseconds as shown in [1]).

Similarly, 34,572 ASes, corresponding to 82.5% of total ASes, have less than 50 /24 addresses. All of those ASes (networks) provide an anonymity set of less than 12,700, just over 0.0008% of the anonymity set of United States, and only under 0.57% of the anonymity set of an average city (8,804 /24 addresses). Those ASes collectively have just under 300,000 /24 address, corresponding 3.7% of

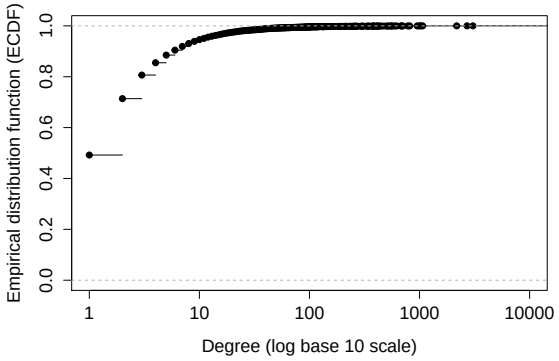


Fig. 9. AS-level degree distribution of networks.

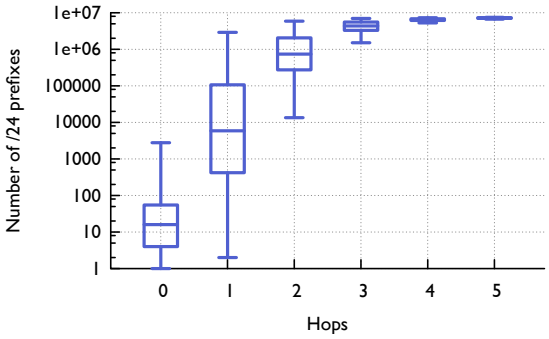


Fig. 10. Reachable /24 addresses for varying hop counts.

the mapped addresses (1.85% of the total IP space), indicating the significance of the problem even when considering an upper bound on the size of a network.

Second, we note that the attack explained in this paper, including the the scenario highlighted in Fig. 2, for any graph of networks with degree at most 3. This is, 1) a node of degree 1 would be a stub network, and user profiling within the network would be possible, 2) a network of degree 2 would be a bridge network, and user profiling within the network would be possible, and 3) a network with degree 3 would be a bridge network, and both user and business intelligence profiling would be possible, per the scenario in Fig. 2. Fig. 9 shows the degree distribution of AS-level networks, based on the most recent topology in [2]. In this figure, we see that more than 50%, 20% and 10% of the networks on the Internet have degree of 1, 2, and 3, respectively, highlighting the potential of the attack. Furthermore, more than 72%, 40% and 23% of the respective networks have less than 10 /24 network addresses associated with them, respectively. Those networks, collectively correspond to more than 2% of the IP space, and about 45% of the total networks on the Internet today.

Traversing multiple autonomous systems while retrieving contents is not a far-fetched assumption. To this end, we study how multiple ASes on the path of the contents between a user and the origin server affect privacy (the anonymity set) as shown in Fig. 10. For that, we calculate the total number of /24 addresses associated with each AS, and plot the distribution as in the first box plot. As reachable ASes from that AS are added with one hop, we add the total number of /24 within reach, and plot the next box plot (first hop). We repeat the process until all ASes are reachable from each source AS. We find that when the number of hops is only two, the minimum total number of reachable /24 addresses is more than 10,000, while the median is about 1,000,000, just less than 1/8th of the ideal total anonymity set. Through our previous timing measurements, we established that it takes only a few hops to move from an AS to another. In conclusion, this highlights that the added time delay would indeed increase the anonymity of the requesting users and thwart the adversary.

5 Discussion

Our protocols and the attack rely on several assumptions. In the following, we argue that those assumptions are not far-fetched, and the attack is severe to the operation of ICN. We further argue that a simple fix to the problem using intelligent caching, while shares similarity with our solution, does not prevent the attack and further degrades the utilization of the network and usere experience.

5.1 Rationale of Assumptions

One of our 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. On the other hand, the suggested solution in [3], while on average would perhaps provide similar gain, would penalize entire requests with with an artificial cache-miss effect.

Another assumption we make is that both an adversary and a benign user are residing behind the same router, and are 1-hop away from each other. On the other hand, if they 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 Fig. 5) have small RTTs, the adversary has to have a very sensitive measurement 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 infrastructure, 2-hop away users could likely be hidden in a large enough anonymity set (cf. §4.2.4). This makes it hard for the adversary to pinpoint a smaller set of users as the potentially the requesters of the contents.

5.2 Threat Assumptions and Their Relaxation

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 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 of the users who issued them.

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 is not within our attack model, we believe that the second requirement would require collusion of the underlying infrastructure (routers) or much larger number of attackers. Even though the attack is possible in theory, our mechanisms raise the bar greatly for such attack in practice.

5.3 Service Degradation Attacks

A closely related attack that can be applied on our protocols is the quality of service degradation attack explained as follows. Given that our protocols treat every user after the first requester as a potential adversary, the adversary tries to request as many names as possible with the privacy flag indicated. Accordingly, all benign users requesting the same names afterwards will be penalized with additional delay that may degrade the quality of the service. This degradation happens as opposed to the case where the first requester of the contents is a benign user who did not use the privacy flag. Assuming that a given name is requested only once by a benign user, the attacker would achieve his goal.

However, the attack has two practicality issues. First, in order for the adversary to perform the attack successfully, he needs to flood routers with all possible names, and such event would trigger further actions by routers. Second, even when the attack succeeds by a match in the requested names by both the adversary and the benign user, it would have a limited impact. In the worst case, the upper bound of degradation is bounded by d. To compute the exact expected degradation resulting from this attack, we consider four different cases (combines all cases of attack/no-attack and cached/not-cached):

- The name has not been requested before, and the attacker flag triggers caching of the contents in the near-by router. In that case, any further request by a benign user would have a gain of $RTT_h - RTT_d$.
- The name has been requested before, and the attacker flag triggers privacy actions (penalty in time) for subsequent requests. In that case, any further request by a benign user would have a penalty of roughly RTT_d (a negative gain of $-RTT_d$).

Thus, the total gain made by both cases is $RTT_h - 2RTT_d$. for the other cases where there is no attack, the first case's gain—where contents are not requested before—is zero, and the second case's gain would be roughly RTT_h. Summing both cases would result in RTT_h gain. By putting both cases together, the attacker's achieved degradation for two requests is $RTT_h - (RTT_h - 2RTT_d) = 2RTT_d$ (averaging RTT_d per request). Accordingly, the adversary has a small incentive in trying the degradation of service attack mentioned above.

5.4 Shortcomings of Intelligent Caching

As discussed in §1.2, one potential approach to possibly prevent the attack is to disable caching of certain contents [16], [29], or to cache contents at certain locations in the network that are far from the router that is the serving contents to benign user and adversary alike [12]. While this gives the same the impression that the content is located far from the user and the adversary, the solution has two shortcomings from the security and performance perspectives.

Security-wise, the solution is vulnerable to probing: an adversary can issue multiple requests of the same content and establish that the content is not being cached according to the original caching mechanism of the ICN, for a reason or another, and likely due to the privacy modifications imposed on the caching mechanism.

Performance-wise, this solution penalizes legitimate users when requesting contents by imposing additional delays on the time needed for serving their requests. This is, every request by any user would require transmitting contents from the location where the contents are cached to the edge router connecting the user to the rest of the network. This overhead is then multiplied by the number of requests associated with each named content, and may degrade the overall utilization of the network. While in our solution we preserve the universal caching mechanisms in ICN. To this end, while both intelligent caching and our solution use similar ideas, they are entirely different in their end goals.

5.5 Timing Attacks on the Timing Protocols

A potential attack on our protocol is by having a large number of observation data to be able to tell the difference between the real delay's probability distribution

and the artificial delays generated by our protocols. In principle, the attack assumes that the two observations are driven from a distribution with different characteristics. While this is easily the case for systems where the designer has no control over the generated delay [35], our experience indicates that a carefully generated delay based on an understanding of today's delay characteristics can eliminate the attack. Second, such attack would only possible in theory, where the interplay between the caching behavior and time distribution, and the artificial delay distribution would add to the confusion of the adversary. Third, while the attack would perhaps enable the adversary to know whether caching is done or not, it would not tell him which user (or set of users) are interested in the contents or how far they are from him or the cache. In particular, our delay generation algorithm need not to be sophisticated, or to consider the day of the week, or time of the day to provide reasonable guarantees: the end goal of the algorithm is to only put the adversary under the impression that contents are fetched from a cache located multiple hops away.

5.6 Overhead Evaluation

Evaluating the overhead at routers and APs would depend 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.

However, one can establish a relative ranking of the different protocols. We notice that the first protocol, which maintains per-user states in the router, requires the largest overhead in relation with other protocols, whereas the last protocol that moves the per-user states to access points requires the least amount of overhead at the router side. The second protocol, which stores per-face states at each router requires similar overhead to that of the third protocol, but does not require states to be stored at the access points, so in total this protocol has the least overhead among the three protocols. We bear in mind that, however, each protocol provides a different level of privacy, granularity and guarantees, see the discussion in the introduction of §3. Thus, the overhead is only one aspect of comparison.

Abstractly, assuming that F is the set of faces, U is the set of potential users using ICN, N is the potential set of names of contents requested via ICN, and A is the set of access points connected to a router via a given face, the storage overhead (at each entity in the network) in each protocol is shown in Table 2. Notice that the overhead at each access point in the third protocol (§ 3.3) is much less than the overhead in the first protocol; the number of access points (to which the overhead is negatively proportional) is orders of magnitude larger than the number of the routers in the ICN.

TABLE 2
An overhead comparison between the different protocols at each entity in the network.

Protocol	Router	Access point
§3.1	$\sim (\|N\| + \|U\|)$	—
§3.2	$\sim (\|F\|)$	—
§3.3	$\sim (\|F\|)$	$\sim [(\|U\| + \|N\|)/\|A\|]$

We note that the overhead in our protocols can be beyond the capabilities of legacy hardware, such as routers with limited memory that cannot hold the states maintained for privacy preservation. To this end, a potential deployment of our protocols is using a software router [33], rather than a hardware one, which would also benefit from the flexibly emerging networking paradigms, such as software defined networks, provide as a capability to the problem at hand.

6 Related Work

With the exception of the concurrent work discussed in §1.2 (in [3] and [29]), this work is the first to address the problem at hand and to provide mitigations to it. Our work, however, relies on the timing channel for its operation, which found use in the web caching (as discussed in §1.2) and the anonymity communities. In the anonymity community, timing is used as a major mechanism for fingerprinting users [23]. For example, timing difference in port scanning responses is used as a side channel for fingerprinting users in the context of Tor by an off-path adversary [20]. However, the relevant solution in [20] is analogous to making names (addresses) unpredictable, which is limited as admitted by the authors of [20], and inapplicable to ICN as discussed in §1.2 (suggested in [29] concurrently with [20])

Future Internet Architectures: Other architectures that date prior to the introduction of CCN [24] and NDN [45] include TRIAD [21] (the work that pioneered the concept of ICN), ROFL [11], and DONA [28]. Other recent architectures include XIA [6], [22], CONET [14], DACON [27], and SCION [47] (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 [4].

Caching in Content Centric Networks: Caching has enjoyed a cornerstone position in the ICN research community, since it is one of the main features that ICN

advocates as an advantage over the current Internet design. The main motivation of such caching algorithms introduced in the literature is performance, rather than privacy. Most of the schemes introduced in the literature for caching aim to reduce the RTT and improve users experience by maximizing the cache hit and minimizing cache misses. Examples of the prior literature on caching in ICN to address these issues include the work in [12], [13], [26], [34], [37], [41], [43].

Security and Privacy in Content Centric Networks: In [44], secure naming system has been proposed. Named-based trust and security protection mechanisms are introduced in [46]. Different naming conventions in ICN architectures and their security features are discussed in [17]. A privacy-preserving contents retrieval in ICN (that assumes the origin server is dishonest) is proposed in [9]. A diverse array of security mechanisms for ICN is introduced in [25]. A closely related architecture that makes accountability as a first-order property, named AIP, is in [7] (which shares similarities with the naming in [10]). Arguments on the benefits of ICN and future Internet design in general are in [19], [40]. Finally, a critique of caching mechanisms, as suggested in NDN is introduced in [18].

7 Concluding Remarks

In this paper we have introduced an attack on content access privacy that is applicable to several ICN architectures. 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. 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.

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 [12], [13], [26], [43], 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] —. Ip latency statistics. http://www.verizonenterprise.com/about/network/latency/, July 2014.

[2] —. Ipv4 and ipv6 as core: Visualizing ipv4 and ipv6 internet topology at a macroscopic scale. http://www.caida.org/research/topology/as_core_network/, July 2014.

[3] G. Acs, M. Conti, P. Gasti, C. Ghali, and G. Tsudik. Cache privacy in named-data networking. In *IEEE ICDCS*, July 2013.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less pain, most of the gain: Incrementally deployable icn. In *ACM SIGCOMM*, pages 147–158, 2013.

[16] E. W. Felten and M. A. Schneider. Timing attacks on web privacy. In *ACM Conference on Computer and Communications Security*, pages 25–32, 2000.

[17] 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.

[18] 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.

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

[20] Y. Gilad and A. Herzberg. Spying in the dark: Tcp and tor traffic analysis. In *Privacy Enhancing Technologies*, pages 100–119. Springer, 2012.

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

[22] 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.

[23] N. Hopper, E. Y. Vasserman, and E. Chan-Tin. How much anonymity does network latency leak? *ACM Transactions on Information and System Security (TISSEC)*, 13(2):13, 2010.

[24] 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.

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

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

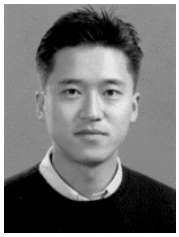
[27] D. Ko, K. Cho, M. Lee, H. Kim, T. T. Kwon, and Y. Choi. Decentralized and autonomous content overlay networking (dacon) with wifi access points. In *Proceedings of ACM CFI*, pages 18–24, 2010.
[28] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM CCR*, 37(4), 2007.
[29] T. Lauinger, N. Laoutaris, P. Rodriguez, T. Strufe, E. Biersack, and E. Kirda. Privacy risks in named data networking: what is the cost of performance? *Computer Communication Review*, 42(5):54–57, 2012.
[30] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable onion routing with torsk. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 590–599. ACM, 2009.
[31] A. Mohaisen, X. Zhang, M. Schuchard, H. Xie, and Y. Kim. Protecting access privacy of cached contents in information centric networks. In *ACM CCS*, pages 1001–1003, 2012.
[32] A. Mohaisen, X. Zhang, M. Schuchard, H. Xie, and Y. Kim. Protecting access privacy of cached contents in information centric networks. In *ACM ASIA CCS*, May 2013.
[33] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker. Extending networking into the virtualization layer. In *Hotnets*, 2009.
[34] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi. Transport-layer issues in information centric networks. In *ACM SIGCOMM ICN*, 2012.
[35] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper. Routing around decoys. In *ACM CCS*, pages 85–96, 2012.
[36] I. Seskar, K. Nagaraja, S. Nelson, and D. Raychaudhuri. Mobilityfirst future internet architecture project. In *Proc. of ACM AINTEC*, 2011.
[37] 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.
[38] The CCNx Project. CCNx. https://www.ccnx.org/, July 2012.
[39] Traceroute. Traceroute. http://www.traceroute.org/, July 2012.
[40] D. Trossen, M. Sarela, and K. Sollins. Arguments for an information-centric internetworking architecture. *ACM CCR*, 40(2), 2010.
[41] 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.
[42] 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.
[43] 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.
[44] W. Wong and P. Nikander. Secure naming in information-centric networks. In *Proceedings of ACM ReARCH*, pages 12:1–12:6, 2010.
[45] 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.
[46] 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.
[47] 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.

Aziz Mohaisen obtained the M.S. and Ph.D. degrees in Computer Science from the University of Minnesota, both in 2012. He is currently a Senior Research Scientist at Verisign Labs. Previously, he was a Member of Engineering Staff at the Electronics and Telecommunication Research Institute, a large research and development institute in South Korea. His research interests are in the areas of networked systems, systems security, data privacy, and measurements. He is a member of IEEE and ACM.

Hesham Mekky received his B.Sc. in Computer Science from Alexandria University, Egypt in 2007, and M.Sc. in Computer Science from University of Minnesota in 2013. He is currently a Ph.D. candidate of Computer Science at University of Minessota. His research interests include networking, security, and privacy. He is a student member of IEEE.

Xinwen Zhang received the PhD degree in information security from George Mason University, Fairfax in 2006. He is currently a Principal Engineer and Director of R&D, Samsung Telecommunications America (Samsung Mobile). His research interests include security policies, models, architectures, and mechanism in general computing and networking systems. His recent research focuses on secure and trusted network infrastructure, cloud computing, and mobile platforms and systems. He is a member of the IEEE.

Haiyong Xie (S'05–M'09) received the B.S. degree from the University of Science and Technology of China, Hefei, China, and the Ph.D. degree in computer science from Yale University, New Haven, CT. He proposed the idea of proactive provider participation in P2P (aka P4P, coordinating network providers and peer-to-peer applications), laid the theoretical foundations, designed and implemented the novel P4P network architecture, and led and conducted original research and large-scale tests on P4P. Encouraged by and based upon his research and results on P4P, the P4P Working Group (P4PWG) was formed to promote academic studies and industrial adoptions of P4P. He was the Principal Researcher for the P4PWG and Distributed Computing Industry Association. He also has conducted research in general computer network areas including network traffic engineering, enterprise network traffic optimization, routing in overlay networks, and network equilibria. He is currently a Principal Research Scientist at US Innovation Center, Huawei Research Labs - USA.



Yongdae Kim is a professor in the Department of Electrical Engineering at KAIST. He received PhD degree from the computer science department at the University of Southern California under the guidance of Gene Tsudik. Prior to join KAIST, he was a faculty member in the Department of Computer Science and Engineering at the University of Minnesota - Twin Cities. He received NSF career award and McKnight Land-Grant Professorship Award from University of Minnesota in 2005. Currently, he is serving as a steering committee member of NDSS (Network and Distributed System Security Symposium). His current research interests include security issues in various systems such as cyber physical systems, mobile/ad hoc/sensor/cellular networks, social networks, storage systems, and anonymous communication systems.