Protecting Access Privacy of Cached Contents in Information Centric Networks

Abedelaziz Mohaisen
Verisign Labs, VA, USA

Xinwen Zhang
Huawei Technologies, CA, USA

Max Schuchard
University of Minnesota, MN, USA

Haiyong Xie
Huawei Technologies and USTC, China

Yongdae Kim
KAIST, Daejeon, South Korea

ABSTRACT

In 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 by other users to be fulfilled quickly. However, the way ICN caching works 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.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General – *Security and Protection*; C.4 [**Performance of Systems**]: Design studies

Keywords

Information centric networks, privacy, side channel attacks, caching.

1. 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) [11] and named data network (NDN) [19], 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 [11]. 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

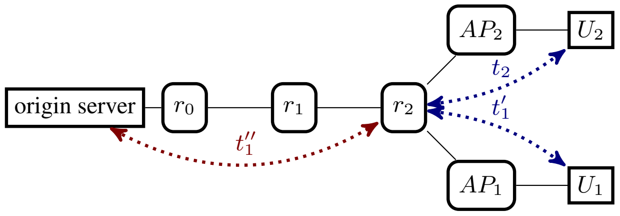

Figure 1: Toy example of timing attack in ICN.

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 [11, 19]. However, this universal caching mechanism poses a great privacy risk: the time difference between a response of cached, when compared to uncached content, can be used as a side channel to infer whether a near-by user has previously requested that content.

For example, 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) 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 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 may require additional side information, an attack like the one described above 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 easily available information. For example, an adversary launching a business intelligence attack might be interested in knowing what contents are being retrieved by a competing company, rather than by individual users. This attack would be possible if the adversary is co-located with that company behind an edge router, and using the above technique.

Shortcomings of Simple Solutions. Simple solutions cannot prevent the attack. For example, a user can mark a content object with a privacy flag to disable in-network caching. However, this will degrade the quality of experience of other users. Also, an adversary can send two consecutive requests of the same name to infer routers' behavior. In the first request, and assuming caching is enabled, the requested data will result in data caching The second request in that case will result in cache-hit, and the content will be served to the adversary quickly. However, if caching is disabled, the second request will result in a delay close to the delay in the first request, from which the adversary can infer that caching is disabled, and that another user has likely used the privacy flag.

Other solutions to the attack by "intelligent caching" based on router-to-user distance have their limitations. To do that, a router has to either know the user's location in advance or have the location provided at the time of caching. While the first approach requires partial topology information that can easily exceed typical routers' resources, the latter one is vulnerable to misuse; to negatively impact the users' experience the adversary can flag any content from an arbitrary location so as to disable caching.

1.1 Contributions

We make the following contributions:

- 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 [2]. We disclaim the originality of the attack in its general form but claim its suitability and applicability to ICN.
- 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. 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.

1.2 Organization

The organization of the rest of this paper is as follows. In Section 2 we review the preliminaries and terminologies used in this paper. In Section 3 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 4 we present our simulation results to validate the attack and evaluate the performance of our defense 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 point out our future work.

2. PRELIMINARIES AND TERMINOLOGY

In ICN, contents are fetched by their names [11]. 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.

2.1 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 (see Section 5). 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 §1 for such scenario). Finally, the attack discussed in this paper is applicable to CCN, and to a lesser extent to other proposals [5, 10, 16].

2.2 Design Goals and Privacy Definition

The design goals in this paper is to protect the privacy of the users fetching contents using the ICN at reasonable cost. We use the classical definition and meaning of the privacy as "anonymity": the adversary should not be able with his reasonable resources (as assumed in section 2.1) to pinpoint the user fetching such contents among a finite number of users within his proximity. To end, we definite the anonymity set of a user fetching the contents as the number of users at an less than or equal distance from the adversary to that user, who could potentially be fetching that same contents. To this end, we outline the following design goals. (i) *Protect the user privacy*: the main design goal in this work is to defend an adversary from inferring users' access patterns to contents fetched and distributed using ICN (details below). The privacy is defined as anonymity, and increasing the anonymity set of the requested traffic so that to make the inference as less accurate as possible would suffice the purpose. (ii) *Cost effective:* the modifications and overhead for providing privacy to the access pattern of the users should not represent a great overhead in relation with the operational overhead of the ICN (used for routing or caching). Furthermore, the protection mechanisms should not generate an excessive amount of communication overhead (in the form of bits on wire). (iii) *Minimal change to existing ICN protocols:* ideally, we want our solutions not to alter the way caching and routing operate in the ICN.

3. PROTECTION MECHANISMS

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

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

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

3.1 The "Vanilla" Approach

The vanilla algorithm to prevent timing attacks on privacy in ICN is described in Algorithm 1. 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—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 Fig. 1), its edge router (r_2 in Fig. 1) maintains $\varphi(u, n) : U \times N \to INT$, where U, N, and INT are the sets of users, content names, and integers, respectively. $\varphi(u, n)$ indicates the number of times that user u has accessed the content name n. At the beginning, assuming benign user U_1 first generates interest *Ints* $= (U_1, n, \mathsf{pmode}, ts_0)$ 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 [11] 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 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 states are stored and maintained in each router, and decisions to preserve privacy are made upon those states. Unlike Algorithm 1, each router stores $\varrho : F \times N \to INT$, where F is the set of faces. $\varrho(f, n)$ indicates the number of times that content name n is requested from face f. The protocol can be illustrated in Figure 1, 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.

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

```
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.
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 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
        $R$ 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 $u$;
    else
        $R$ returns cached $Data$ to $u$;
    end
end
```

3.3 Low Granularity Approach

The main shortcoming of the approach described in §3.2 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 §3.1. To enable lower granularity in the protocol described 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 aiming to maintain the advantage of both protocols, by maintaining and distributing these states concerning access patterns of individual users at the APs, 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.

4. RESULTS AND ANALYSIS

To understand the potential of the attack and how our designs impact the performance, we perform several measurements on the CCNx prototype [2] in simulated setting. To derive an accurate representation of real-world timings, we feed the simulator with topologies and per-hop RTT traces driven from the current Internet using traceroute [3] to request several websites (origin servers).

4.1 Settings and Timing Data-sets

Our measurements are based on CCNx, an open source system that implements CCN. CCNx implements both the communication operations (naming and data conventions) and security operations (using OpenSSL), for signature generation and verification.

Because no ICN design is deployed yet, we lack real-world traces of RTTs for real ICN. However, designs like CCN suggest operating on top of IP, making IP timings relevant for experiments. To this end, we instrument the CCNx simulator with 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 [1]. 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 origin server is shown in Figure 2. Complementary CDF for smaller range of RTT ($RTT_i \leq 1$) is shown as a small graph within the CDF in

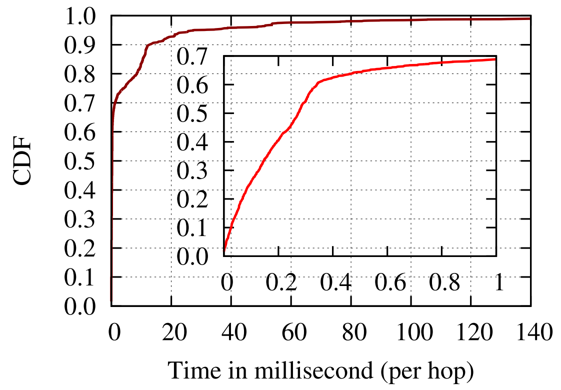

Figure 2: An empirical CDF of the per-hop RTT for alexa-100.

Figure 2 (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 and the first hop delay (used for validating the attack).

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, we consider 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 hop—notice that $RTT_1 = 2td_0$ in our protocols) for each of the 24 sites is shown in Figure 3. 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 plot of the normalized RTT up to the origin server is in Figure 4.

4.2 Results

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 Figure 4, 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

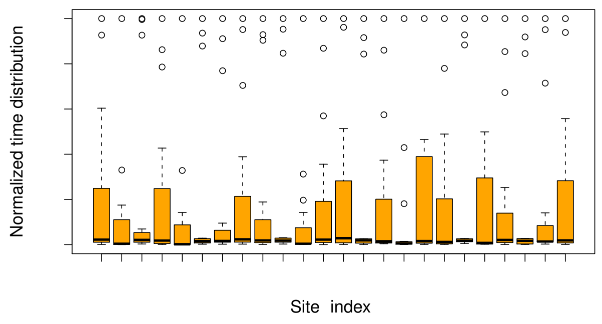

Figure 3: The normalized per-hop total RTT for 24 sites.

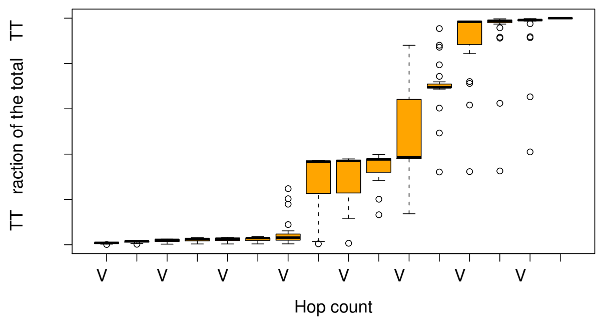

Figure 4: The normalized RTT up to the given hop count.

profiles in Figure 3 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.

How defenses impact the performance. 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 §4.1), 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, we consider the maintained RTT gain (normalized) as $1 - (td(n)/td_x) \approx 1 - RTT_d^t$. We compute this quantity for the min, max, mean, and median RTT_d^t of the different sites, for different d values.

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 5. 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,

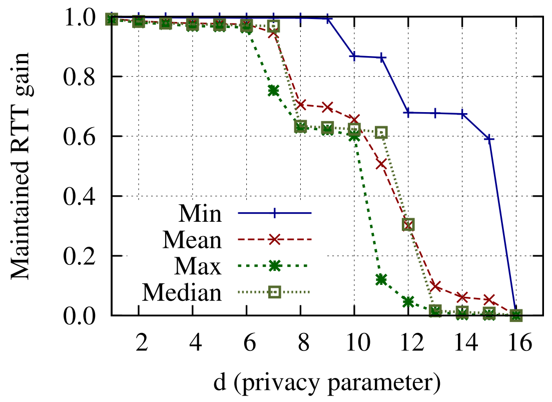

Figure 5: Maintained RTT gain for varying d values.

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 to provide to users.

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 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 0.92 at average.

Overhead evaluation. The overhead depends greatly on how often contents are flagged for privacy. 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.

5. DISCUSSION

We assume that the adversary and the benign user are residing behind the same router, and are 1-hop away from each other. Thus, 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 4) have small RTTs, the adversary must have sensitive measurements capability at the microsecond level to be able to tell if the user is 2, 3, or 4 hops away with confidence. Second, 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.

We also assume 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 is only applicable to the first protocol. In the two latter protocols, requests have to go through the same face, thus they will be considered as if they are from the same entity, regardless to the users who issued them.

A final attack is what we coin the "intersection attack", in which two geographically distributed attackers collude to infer if a piece of contents is cached or not. For example two nodes that are 3 hops away on the path of contents receiving the same time measurements can infer the manipulation by the router by contradicting their measurements . However, in order for this attack to work, the attackers need to: 1) be geographically distributed, and 2) know in advance the path benign requests have traversed. 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 the attack in practice.

6. RELATED WORK

Concurrent to our work, [14] pointed out the attack under different caching policies, but falls short in not providing any workable solutions to it. Caching has been widely investigated, although motivated by the performance rather than privacy. Examples of the prior literature include the work in [8, 13, 15].

Security and privacy in ICN have been discussed in several recent works. In [18], secure naming system has been proposed. Named-based trust and security protection mechanisms are introduced in [20]. Different naming conventions in ICN architectures and their security features are discussed in [9]. 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 [12]. 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 [17].

7. CONCLUSION

We have introduced an attack on content access privacy that is applicable to several ICN architectures, including the 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. Our future work will include looking into how different caching polices will affect our attack and countermeasures.

Acknowledgement

H. Xie is supported in part by Grant No. WK0110000014, by NSFC Grant No. 61073192 and by 973 Program Grant No. 2011CB302905. The last author was supported by the KCC (Korea Communts Commission), Korea, under the R&D program supervised by the KCA (Korea Communications Agency), KCA- 2013-12911-05003.

8. REFERENCES

[1] Alexa's top sites. `http://bit.ly/2FzvdZ`, 2012.

[2] CCNx. `https://www.ccnx.org/`, July 2012.

[3] Traceroute. `www.traceroute.org`, 2012.

[4] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable internet protocol (AIP). In *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 *ACM ICN*, 2011.

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

[8] W. Chai, D. He, I. Psaras, and G. Pavlou. Cache "less for more" in icns. *NETWORKING*, 2012.

[9] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in content-oriented architectures. In *ACM ICN*, 2011.

[10] 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 *NSDI*, 2012.

[11] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. Braynard. Networking named content. In *ACM CoNEXT*, 2009.

[12] J. Jeong, T. T. Kwon, and Y. Choi. Host-oblivious security for content-based networks. In *ACM CFI*, 2010.

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

[14] 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.

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

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

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

[18] W. Wong and P. Nikander. Secure naming in information-centric networks. In *ACM ReARCH*, 2010.

[19] 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.

[20] X. Zhang, K. Chang, H. Xiong, Y. Wen, G. Shi, and G. Wang. Towards name-based trust and security for content-centric network. In *IEEE ICNP*, 2011.