Cache Privacy in Named-Data Networking

Gergely Acs[*,#], Mauro Conti[†,#,+], Paolo Gasti[‡,#], Cesar Ghali[§], Gene Tsudik[§]
[*]*INRIA*
[†]*University of Padua*
[‡]*New York Institute of Technology*
[§]*University of California, Irvine*

***Abstract*—Content-Centric Networking (CCN) is an alternative to host-centric networking exemplified by today's Internet. CCN emphasizes content distribution by making content directly addressable. Named-Data Networking (NDN) is an example of CCN being considered as a candidate next-generation Internet architecture. One key NDN feature is router-side content caching that optimizes bandwidth consumption, reduces congestion and provides fast fetching for popular content. Unfortunately, the same feature is also detrimental to privacy of both consumers and producers of content. As we show in this paper, simple and difficult-to-detect timing attacks can exploit NDN routers as "oracles" and allow the adversary to learn whether a nearby consumer recently requested certain content. Similarly, probing attacks that target adjacent content producers can be used to discover whether certain content has been recently fetched. After analyzing the scope and feasibility of such attacks, we propose and evaluate some efficient countermeasures that offer quantifiable privacy guarantees while retaining key features of NDN.**

I. Introduction

Today's Internet has become a *de facto* public utility, with a large percentage of the world's population relying on it for numerous activities. However, despite its unparalleled success and popularity, current Internet's IP-based architecture is rapidly aging. Anticipating the need for the next-generation Internet architecture, a number of research efforts have sprung up in recent years [23].

One key motivator for a new Internet architecture is the fundamental shift in the nature of traffic: from mainly low-bandwidth interactive (e.g., remote log-in) and store-and-forward (e.g., email) nature of the early Internet to web-dominated Internet of today. At the same time, massive and ever-increasing amounts of content are being constantly produced and consumed (distributed) over the Internet. This transpires over social networks such as Facebook, media-sharing sites such as YouTube and productivity services such as GoogleDocs. Consequently, the basics of Internet communication shifted from a telephony-like conversation between two IP interfaces to a consumer who wants content and wants it fast, regardless of where it comes from. This motivates reconsidering Internet architecture.

Content-Centric Networking [7] (CCN) is a recent paradigm for content distribution in large networks. Unlike today's largely host-centric networking (e.g., the Internet), CCN is based on directly addressing content. A consumer requests content by *name* (i.e., expresses *interests* in the content) and the network takes care of finding and returning. The CCN approach moves hosts into the background, while named content becomes a first-class object.

One example of CCN is Named-Data Networking (NDN) – a current research effort considered as a candidate next-generation Internet architecture. NDN is also one of several NSF-funded research projects under the Future Internet Architectures (FIA) program. As part of FIA, NSF emphasizes *"security and privacy by design"*. In today's Internet, security and privacy problems were (and are still being) identified along the way and patches are retrofitted; some haphazardly. In contrast, the NSF FIA program stresses early awareness as well as support for features and counter-measures, from the outset. To this end, security and privacy issues in all FIA projects, including NDN, are being investigated, e.g., [9], [27].

One important NDN feature is router-side content caching. Its goal is to reduce congestion while improving throughput and latency for popular content. In contrast with IP, NDN routers can often satisfy interests using previously forwarded content objects. Consequently, content might be fetched from an NDN router's cache rather than from its original producer.

Motivation. Despite its clear benefits, content caching in NDN also raises a major privacy issue that we summarize below and then discuss, in more detail, in Section III.

Suppose the adversary (denoted by Adv) wants to determine whether a consumer (Alice) recently requested certain content $\mathcal{C}$. If Adv and Alice share a first-hop NDN router R (e.g., they are neighbors who are served by the same ISP), Adv measures round-trip time (RTT) Adv $\leftrightarrow$ R. Next, it requests $\mathcal{C}$ and measures the corresponding RTT. Comparing the two RTTs is sufficient to determine whether $\mathcal{C}$ was retrieved from R's cache.[1] Similarly, suppose that Adv wants to learn whether a content producer (Bob) has been recently asked for, and distributed, content $\mathcal{C}$. Assuming that Adv and

[#]Work performed in part while at UC Irvine.
[+]Mauro Conti was supported by Marie Curie Fellowship PCIG11-GA-2012-321980, funded by the European Commission for the PRISM-CODE project, and by the University of Padua Researchers' Mobility grant 2012. This work has been partially supported by the TENACE PRIN Project 20103P34XC funded by the Italian MIUR.
The work of Paolo Gasti, Cesar Ghali and Gene Tsudik was supported by the NSF under project CNS-1040802 – "FIA: Collaborative Research: Named Data Networking (NDN)".

[1]Clearly, there might be other users, besides Alice and Adv. However, this would not change the nature of the attack.

Bob are separated by at least one NDN router, Adv measures or estimates Adv $\leftrightarrow$ Bob RTT and then requests $\mathcal{C}$. If the latter RTT is lower than the former, Adv concludes that $\mathcal{C}$ was fetched from some NDN router cache and, therefore, at least one consumer recently requested it. Furthermore, a combination of these two attacks can be used to learn whether two parties (Alice and Bob) have been recently, or still are, involved in a two-way interactive communication, e.g., voice or SSH.

These attacks do not require Adv to have any special privileges: the interaction between Adv and NDN routers is normal. It might appear that the Adv can learn the same information by simply eavesdropping on Alice or Bob. However, eavesdropping requires real-time presence by Adv who must also be physically near the victim (e.g., the same ethernet segment, wired or wireless). Whereas, aforementioned attacks require neither real-time presence nor proximity.

Roadmap. In this paper, we explore privacy issues in NDN router-side content caching. We argue that access to router caches allows users to obtain information about their neighbors' incoming and outgoing content traffic. Next, we show, via experiments with current NDN public-domain prototype, that these privacy threats are realistic. We then propose several simple countermeasures that, predictably, offer tradeoffs between privacy and cost. We also introduce a formal framework for privacy evaluation of shared router caches and evaluate proposed countermeasures within this framework.

Organization. Section II overviews NDN. We show how to implement and perform privacy attacks in Section III, also supporting our claims with experiments run on the current NDN testbed [21]. Then, adversary and privacy models are described in Section IV. Next, Section V discusses proposed countermeasures. Experimental evaluation and real-world impact of these countermeasures are presented in Section VII. Finally, Section VIII reviews related work and the paper concludes with some items for future work in Section IX.

II. NDN Overview

NDN ([20], [15]) is a network architecture based on named content. Rather than addressing content by location, NDN refers to it by human-readable names. A content name is composed of one or more variable-length components that are opaque to the network. Component boundaries are explicitly delimited by "/" in the usual representation. For example, the name of a CNN news content for May 20, 2013 might look like: `/cnn/news/2013may20`. Large pieces of content must be split into fragments: fragment 137 of Alice's YouTube video could be named: `/youtube/alice/video-749.avi/137`. Since NDN's main abstraction is content, it is not possible to directly address "nodes" (hosts, routers), albeit, their existence is assumed.

NDN communication adheres to the *pull* model: content is delivered to consumers only upon explicit request. A consumer requests content by sending an *interest*. If an entity (a router or a host) can "satisfy" a given interest, it returns the corresponding *content*. A content named X is never forwarded or routed unless it is preceded by an interest for name X.[2] Interest and content are the only types of packets in NDN.

Each NDN node (router or host) maintains the following three components [6]:

- *Content Store* (CS) – cache used for content caching and retrieval. (From here on, we use the terms *CS* and *cache* interchangeably).
- *Forwarding Interest Base* (FIB) – routing table of name prefixes and corresponding outgoing interfaces (to route interests).
- *Pending Interest Table* (PIT) – table of currently not-yet-satisfied (pending) interests and a set of corresponding incoming interfaces.

When a router receives an interest for name X, and there are no pending interests for a matching name in its PIT, it forwards the interest to the next hop(s), according to its FIB. For each forwarded interest, a router stores some amount of state information, including the name in the interest and the interface on which it arrived. However, if an interest for X arrives while there is already an entry for the same content name in the PIT, the router collapses the present interest (and any subsequent ones for X) storing only the interface on which it was received. When content is returned, the router forwards it out on all interfaces on which an interest for X has arrived and flushes the corresponding PIT entry. Since no additional information is needed to deliver content, an interest does not carry a "source address".

Each NDN router provides content caching and its cache size is determined by local resource availability. Routers unilaterally determine what content to cache.

At the first glance, NDN seems inherently safe against cache privacy attacks: if multiple consumers share the same NDN router's cache, Adv cannot determine exactly *which* or *how many* requested particular content, as long as the content does not expose consumer-identifying information. Thus, consumers seem to benefit from some form of k-anonymity. However, for many types of traffic (e.g., email, instant messaging, voice) consumer's identity can be determined by examining the content and/or its name. Furthermore, since all content objects in NDN are signed, content producer can be easily identified by its signature. Finally, even k-anonymity may be insufficient if Adv simply wants to determine whether *any* consumer requested a particular content. We also note that content encryption offers limited benefits. Encryption conceals the "payload" of content objects. However, NDN names cannot be encrypted, since doing so would prevent content objects from being routed.[3] In general, since Adv aims to determine whether certain named content has

[2] Strictly speaking, content named $X' \neq X$ is a match – and therefore can be sent in response to – an interest for X if and only if X is a prefix of X', e.g., `/cnn/news/2013may20` matches `/cnn/news/`.

[3] Onion routing over NDN [9] has been investigated. It allows parties to encrypt content names. However, it incurs a heavy performance burden.

been sent and/or received, encryption is basically ineffective.

Based the above discussion, there is an inherent conflict in NDN between utility of router-side caching and privacy for both producers and consumers of content. The resulting privacy problem and potential countermeasures are the subject of this paper. We believe it is imperative to address (or at least mitigate) them before NDN can be deployed on a large scale.

III. Cache Privacy Attacks

We now describe, in more detail, cache privacy attacks mentioned in Section I. NDN topology for our sample setup is shown in Figure 1. It includes the following entities: (1) user U; (2) NDN router R; (3) content producer P; and (4) adversary Adv. P is reachable by U and Adv only through R. We also assume that only U and Adv are served by R as their fist-hop NDN router, though we will later relax this assumption.

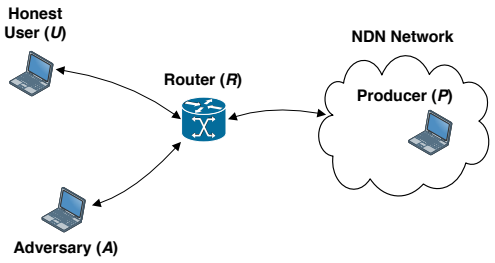


Figure 1. Local network topology

In this topology, Adv can learn whether specific content $\mathcal{C}$ was recently requested by U by probing R's cache. To do so, Adv issues an interest for $\mathcal{C}$ and measures delay d_1 needed to retrieve it. It then picks another (existing) content $\mathcal{C}'$ and requests it twice in succession. The first time, $\mathcal{C}'$ might be fetched from its original producer or from some router's (possibly, R's) cache. However, the second time, $\mathcal{C}'$ is certainly fetched from R's cache. Let d_2 denote the delay for the latter. If $d_1 \approx d_2$, Adv concludes that U recently requested $\mathcal{C}$. Whereas, if $d_1 > d_2$, Adv decides that $\mathcal{C}$ has not been recently requested by anyone from this side of R.[4]

To validate the above scenario, we conducted some experiments. First, P published 1,000 content objects. Next, U, which is directly connected to R via Fast Ethernet, requested all published content objects. This caused all content to be cached by R. Then, from Adv we requested the same contents which were promptly fetched from R's cache. We repeated this experiment 50 times (every time starting with an empty cache for R), and measured average delays for retrieving content from P and R. Our results are in Figure 3(a) show the probability distribution function for the delays. Accordingly, Adv can determine, with probability over 99.9% whether $\mathcal{C}$ is retrieved from R's cache.

We then conducted similar measurements in a WAN topology over the NDN testbed [21], where U and Adv are connected to the same first-hop NDN router R, which is several hops away from both, while P is 3 hops away from R. Figure 3(b) shows the results. Clearly, presence of additional hops increases delay and introduces some variance in measurements. Nonetheless, we can still determine – with probability over 99% – whether $\mathcal{C}$ is retrieved from R's cache.

In our third experiment (again in the NDN testbed setting), P is directly connected to R, while U and Adv are three hops away. Adv's goal was to determine whether $\mathcal{C}$ produced by P was requested recently. To do so, Adv fetches $\mathcal{C}$ twice and measures the delay each time. Results in Figure 3(c) show that Adv can distinguish whether $\mathcal{C}$ is served from R with over 59% probability by probing a single content object.

As mentioned in Section II, large NDN content is split into several (small) content objects. Adv can exploit the correlation between such content objects to improve the accuracy of its guesses in our third experiment. In fact, Adv only needs to determine whether *one* content object has been retrieved. Let **success** denote the event of the adversary successfully determining whether a single content object is fetched from the cache, and **fail** failure to do so. (Hence, $\Pr[\textbf{success}] = 1 - \Pr[\textbf{fail}]$). Since **fail** and **success** are independent for content objects, the overall probability of failure (**FAIL**) can be expressed as $\Pr[\textbf{FAIL}] = (\Pr[\textbf{fail}])^n$. Analogously, $\Pr[\textbf{success}] = 1 - (\Pr[\textbf{fail}])^n$. In our experiment we have $\Pr[\textbf{success}] = 0.59$. ($\Pr[\textbf{fail}] = 0.41$). If a content is split into eight content objects, $\Pr[\textbf{success}] = 1 - 0.41^8 \approx 0.999$.

In probing router caches, Adv can also take advantage of the `scope` field in NDN interests by setting it to "2". This value means that the interest can traverse at most two NDN entities, source included. This way, if Adv receives $\mathcal{C}$ for an interest with `scope` $= 2$, then (regardless of the delay) $\mathcal{C}$ must be in R's cache. However, NDN routers are allowed to disregard this field without significantly affecting NDN functionality.

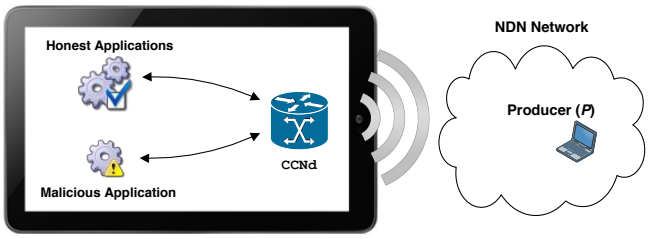
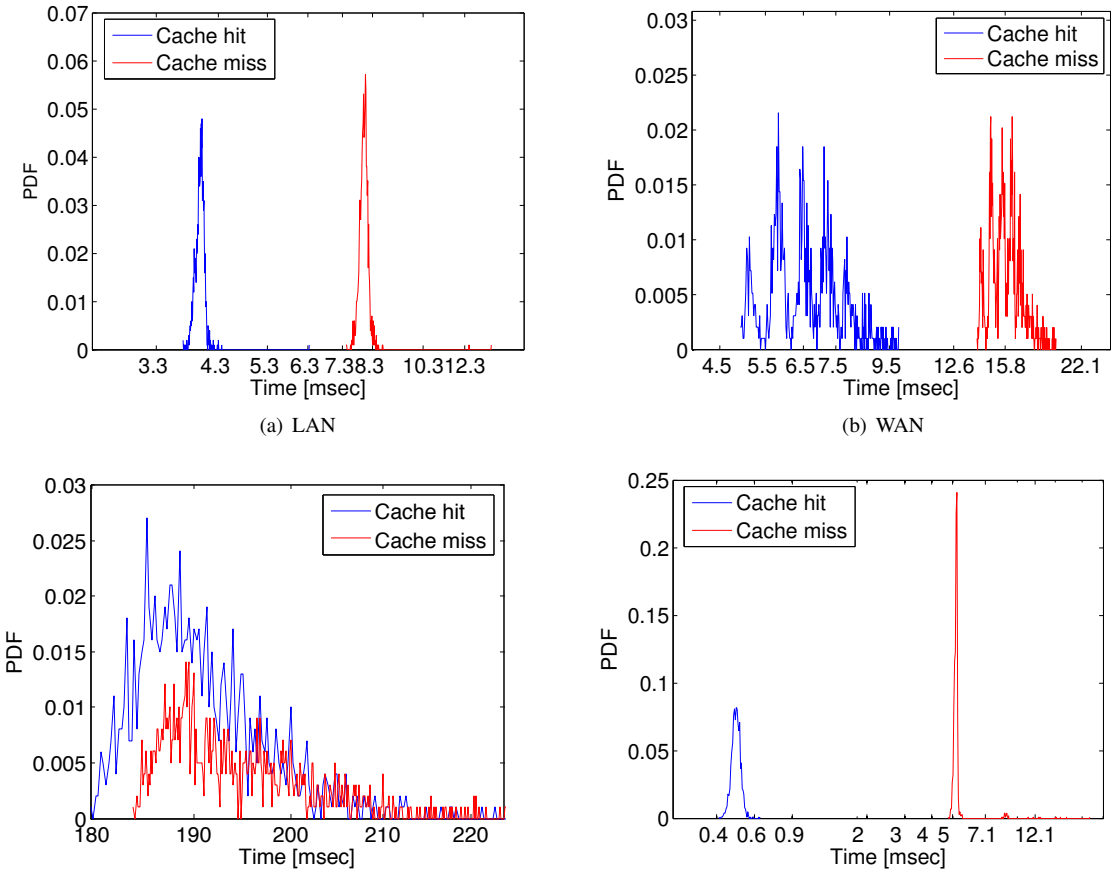


Figure 2. Local network topology

Local Adversary. Attacks described thus far, using the topology in Figure 1, are also applicable to the local cache of a specific NDN node. An NDN node (e.g., a laptop or an Android smartphone [5]) might run multiple applications that have access to the network and produce/consume content. NDN specifications [15], [6] require each NDN node to maintain a local cache. A malicious application, can abuse this cache by employing the same probing techniques

[4]Note that $d_1 < d_2$ is not possible, since we assumed that R is the first-hop NDN router for both U and Adv.



(a) LAN

(b) WAN

(c) WAN, producer privacy

(d) Local Host

Figure 3. Timing Attack Results

described above. Figure 2 illustrates the corresponding topology.

Figure 3(d) summarizes our results in the local-host setting. The difference between cache hits and cache misses is even more evident than in previous experiments, i.e., Adveasily learns information about cached content of other applications.

IV. System, Adversary and Privacy Model

In this section we introduce our system, privacy and adversary models.

System Model. Let $\mathbb{NM}$ and $\mathbb{CN}$ denote the universes of all names and content, respectively. As before, let R be an NDN router. Internal state of R is represented by a function $S : \mathbb{CN} \rightarrow \mathbb{N}$ that, for a given content, represents the number of times it has been forwarded.

$S(\mathcal{C}) = 0$ for all $\mathcal{C}$ that is not in R's cache. We assume that $\mathcal{C}$ can appear in R's cache only if it has been previously forwarded by R.

A cache management algorithm $\mathcal{CM}$ has access to R's internal state and determines what content forwarded by R needs to be cached. $\mathcal{CM}$ also controls how R responds to interests that correspond to cached content. Without loss of generality, we assume that users have access to content only through R, i.e., R is their only choice as a first-hop NDN router. We make no assumptions about how $\mathcal{CM}$ responds to interests that match content in its cache, e.g., $\mathcal{CM}$ is free to ignore its cache altogether for some incoming interests. Finally, we assume that $\mathcal{CM}$ can hide cache hits (e.g., by simply not using its cache) but cannot hide cache misses.

By interacting with R, users are allowed to determine, with some probability, whether a specific content has been forwarded by R (i.e., probing attacks). We model this by a probabilistic algorithm $Q_S : \mathbb{NM} \rightarrow \{0, 1\}$ with access to the router's internal state S. Q_S outputs 1 if some cached content $\mathcal{C}$ matches input name. Otherwise, Q_S outputs 0. After each invocation of Q_S, S transitions to S' such that $S'(\mathcal{C}) = S(\mathcal{C})+1$ and, for all other $\mathcal{C}' \neq \mathcal{C}$, $S'(\mathcal{C}') = S(\mathcal{C}')$.

Adversary Model. The goal of Adv is to learn information about content forwarded (and likely still cached) by R. Since caching algorithm $\mathcal{CM}$ is not secret, Adv can use Q_S to learn private information. In particular, Adv can test whether content $\mathcal{C}$ was recently forwarded by querying $Q_S(\mathcal{C})$.

In our model, Adv can be any NDN entity that requests and receives content. Adv is not allowed to compromise any honest (intended victim) users. Also, Adv does not eavesdrop on communication between R and honest users. (This can be supported by using an encrypted channel

between each user and its closest NDN router). Moreover, Adv cannot compromise R.

Privacy Model. We now turn to privacy definitions. We take advantage of the concept of (ε, δ)-probabilistic indistinguishability [13], [19] – a standard notion to measure indistinguishability of two distributions in privacy-oriented applications.

Definition IV.1 ((ε, δ)-probabilistic indistinguishability)**.** *Two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are (ε, δ)-probabilistically indistinguishable, if we can divide the output space $\Omega = Range(\mathcal{D}_1) \cup Range(\mathcal{D}_2)$ into Ω_1 and Ω_2 such that:*

- *for all $O \in \Omega_1$, $e^{-\varepsilon} \leq \frac{\Pr(\mathcal{D}_1 = O)}{\Pr(\mathcal{D}_2 = O)} \leq e^{\varepsilon}$*
- $\Pr(\mathcal{D}_1 \in \Omega_2) + \Pr(\mathcal{D}_2 \in \Omega_2) \leq \delta$.

Two distributions are "close" if both ε and δ are small. This definition is stronger than the widely used *statistical indistinguishability* since it requires similar probabilities for *each* output in Ω_1. The set Ω_2 contains all "bad" outputs with probabilities in $\mathcal{D}_1$ and $\mathcal{D}_2$ that differ substantially; their ratios can not be bounded by e^{ε} (or $e^{-\varepsilon}$). Intuitively, if $\mathcal{D}_1$ and $\mathcal{D}_2$ represent output distributions of $\mathcal{CM}$ with two different states, then (ε, δ) measures the information that $\mathcal{CM}$ leaks about the states. Any output from Ω_2 typically leak "too much" information: e.g., occurrence of any $O \in \Omega_2$ such that $\Pr(Q_{S_1}(\ell) = O) > 0$ and $\Pr(Q_{S_2}(\ell) = O) = 0$, for the same name ℓ ($S_1 \neq S_2$), may result in privacy breach in practice, as S_1 and S_2 become distinguishable through $\mathcal{CM}$ in that case.

We now define *perfect privacy* with respect to forwarded content. Informally, $\mathcal{CM}$ provides perfect privacy if the way it responds to Q_S queries does not yield any information about the content of the router's cache.

Definition IV.2 (Perfect privacy)**.** *For all names $\ell \in \mathbb{NM}$, subset of content $M \subset \mathbb{CN}$, and pairs of states S_0, S_1 such that $S_0(x) = S_1(x)$ for all $x \in \mathbb{CN} \setminus M$ and $S_0(y) \neq S_1(y)$ for all $y \in M$, $Q_{S_0}(\ell)$ and $Q_{S_1}(\ell)$ are $(0, 0)$-probabilistically indistinguishable.*

The above definition is quite strong since it implies that, if $\mathcal{CM}$ offers perfect privacy, it does not reveal any information about any content previously forwarded by R. We believe that this level of privacy may not be necessary in practice. For this reason, we use the concept of content popularity to relax the above definition: there is no need to conceal the presence of popular content (e.g., Google's home page) from routers' caches, since Adv can safely assume that this content is in caches *without* probing them. Let k be the number of requests above which a content is considered *popular*. We allow the distributions of outputs of Q under two states S_0 and S_1 to be non-indistinguishable, with some probability which depends on k.

Definition IV.3 ((k, ε, δ)-privacy)**.** *For all data names $\ell \in \mathbb{NM}$, subset of content $M \subset \mathbb{CN}$, and pairs of states S_0, S_1 such that $S_0(x) = S_1(x)$ for all $x \in \mathbb{CN} \setminus M$ as well as $S_0(y) = 0$ and $0 < S_1(y) \leq k$ for all $y \in M$ (i.e., S_0 and S_1 only differ on M); $Q_{S_0}(\ell)$ and $Q_{S_1}(\ell)$ are (ε, δ)-probabilistically indistinguishable.*

V. Countermeasures

One trivial and effective countermeasure to all aforementioned attacks is to simply disable router caching altogether. However, this would immediately negate efficient content distribution, which is one of the key benefits of NDN.

Clearly, not all content is private. To avoid the overhead of concealing non-private content, consumers and/or producers need the means to specify which content is sensitive. There are several ways to do so, depending on who decides on content's sensitivity. We outline some examples (the list is not exhaustive):

- **Producer-driven:** a reserved name component, e.g., "`/private/`". If it is present as (for example) the last component of a name, the eponymous cached content is treated as private by the router caching algorithm. Alternatively, a special privacy bit – also set by the producer – in the content header could achieve the same effect. A consumer issues interests as usual, possibly remaining oblivious to content being private.
- **Consumer-driven:** a special privacy bit in interests, set by the consumer. The corresponding content, when cached by the router is marked accordingly and is treated as private. The producer may or may not pay any heed to the privacy bit, partly depending on whether the interest propagates all the way to the producer.
- **Mutual:** content is referred to by an unpredictable name. In other words, if content is private, both consumer and producer refer to it by a name that contains a unique and hard-to-guess component, ideally derived from some shared secret. One attractive feature of this approach is its opaqueness – routers need not be aware of its existence.

These three approaches are not mutually exclusive, e.g., even if $\mathcal{C}$ is not marked (or named) as private by its producer, it can be requested as private by a consumer.

The above alternatives only correspond to the means of marking private content. However, they do not imply or include any actions by NDN entities (particularly, routers) that need to be taken upon encountering such private content. To this end, in the rest of this section we introduce techniques that inhibit Adv from extracting meaningful information about forwarded private traffic from routers' caches.

In designing countermeasures, we consider two types of network traffic: *interactive* and *content distribution*. The first represents synchronous communication between two or few parties, e.g., VoIP and remote shell. This type of traffic is characterized by the requirement for low-latency and continuos interaction. In other words, communicating parties continuously play the roles of both producer and consumer. Whereas, multimedia data delivery, live broadcasts and delivery of web pages are examples of content distribution traffic. Our rationale for distinguishing among these two traffic types in terms of countermeasures is discussed below.

A. Interactive Traffic

While NDN router caching mostly benefits content distribution, it also helps mitigate effects of packet loss in interactive communication [17]. This is because interests re-issued for lost packets can be usually satisfied by content cached closest to the location of actual loss, thereby reducing the delay for re-requested content. For this reason, any privacy-enhancing caching mechanism for this class of traffic should not introduce any additional delay.

At the same time, since interactive content tends to be very time-sensitive, there are hardly any benefits from caching it in routers in the longer term. Specifically, if several users take part in a video-conference, cached stale video frames are of no use to any of them.

We choose to protect this class of traffic using unpredictable names, i.e., the *mutual* approach introduced above. Producers and consumers use a random quantity $rand$ as the last component of the name of each content they create and request, respectively. This requires some coordination between the two (or more) parties involved in the interaction. Regardless of the details, the parties need to agree on a shared secret for seeding a pseudo-random function (e.g., a keyed cryptographic hash, such as HMAC) used to generate content-name-specific $rand$.

We take advantage of our previous assumption that Adv can not eavesdrop on content consumers/producers involved in interactive communication or on traffic over R's incident links (e.g., due to link encryption or lack of physical access). Unpredictable content names inhibit malicious probing of R's cache.[5] At the same time, in the event of packet loss, U can re-issue an interest and still benefit from obtaining requested content from the NDN router closest to the location of packet loss.

As mentioned earlier, this mutual approach to marking private traffic can be combined with producer- and/or consumer-driven approaches. However, that would require router participation.

B. Content Distribution Traffic

Unlike interactive traffic, content distribution does not require any coordination between producers and consumers. Also, comparatively, it benefits a lot more from longer-term router caching. (Consider, for example a YouTube video going viral). Consequently, an ideal privacy approach for content distribution traffic would retain at least some benefits of caching, beyond simple packet loss recovery.

In this setting, neither the number nor identities (or locations) of possible consumers of given content is known in advance. The same goes for timings of their content requests. Thus, the previously described method of using unpredictable content names – which seems well-suited for interactive traffic – is not viable for content distribution. More generally, all *mutual* techniques are essentially ruled out, leaving us with producer- and/or consumer-driven means of marking private traffic. In order to maximize flexibility, we allow both: a producer can mark any of its content as private and a consumer can request any content as private, regardless of how it is marked by the producer.

Based on our discussion thus far, it appears that router involvement in handling private traffic is unavoidable for content distribution. Since low latency is not usually a primary requirement for content distribution traffic, NDN routers can hide cache hits by introducing artificial delays before responding with privacy-sensitive cached content. This strategy retains one important benefit of router caching, i.e., reducing congestion and conserving bandwidth. In other words, bandwidth utilization of NDN remains intact, since this approach does not increase the overall amount of traffic forwarded by any router, as compared to the current NDN architecture. Moreover, if the overall delay introduced by routers is close to the RTT between U and P, the behavior of the network from the U's prospective becomes analogous to the current IP-based Internet.

We now need to consider *which* routers should introduce artificial delays. If all NDN routers independently do so, overall delay for consumers requesting content would likely become unbearable. We believe that a sensible approach is to involve only consumer-facing routers, i.e., those that are most likely to be probed by Adv.[6]

Now, suppose that we introduce constant delay γ, i.e., in case of a cache hit, each consumer-facing R waits for γ milliseconds before returning private-marked content. In case of a cache miss, the artificial delay at R must be the difference between γ and the actual delay for R to receive requested content. Note that, in the latter case, the overall interest-in→content-out delay would still be γ.

This approach is easy to implement and requires a small amount of per-cache-entry state. However, it has a major drawback in that it either penalizes nearby content or sacrifices privacy for far-away content. The former happens if γ is set too high and content with nearby (with respect to R) producers winds up being unduly delayed. Whereas, the latter occurs when requested content is far away (or routed via slow and/or congested links) and the actual delay at R exceeds γ.

It is unclear how to determine the optimal value of γ that would avoid both problems. We therefore consider two alternatives to constant artificial delay:

- *Content-specific delay:* For each privacy-sensitive content $\mathcal{C}$, R stores the original interest-in→content-out delay $\gamma_{\mathcal{C}}$. (In other words, time it took R to obtain $\mathcal{C}$, from either its producer or some other router's cache, the first time.) If an interest for it arrives while $\mathcal{C}$ is in R's cache, R delays replying for $\gamma_{\mathcal{C}}$ msecs.
- *Dynamic delay:* A router can dynamically adjust artificial delay to mimic current behavior of in-network caching for popular content: as the number of interests for a given content grows, so does the likelihood

[5] Routers must not return content that include $rand$ as a name component to interests that do not explicitly express it. For example, content named `/alice/skype/0/`$rand$ should not be returned to interests for `/alice/skype/` even thought it would be a longest-prefix match.

[6] This point deserves further consideration; we defer it to future work.

of it being cached at a nearby router. According to Definition IV.2, the artificial delay must not drop below the actual delay for content located two hops from Adv.

The former is obviously the safer choice for privacy. Albeit, it imposes considerable delays for content that was originally fetched from far-away producers or routers. Whereas, dynamic delay is clearly more responsive to ephemeral traffic patterns, in return for requiring routers to constantly monitor delay and popularity for all content.

Finally, we consider how a router should handle all possible combinations of consumer- and producer-driven content marking. If $\mathcal{C}$ is marked as private by its producer, this setting must be always honored by all consumer-facing routers, even if a consumer requests it without setting the privacy bit in the interest.

Different rules apply to content that is not marked as private by its producer. This type of content is problematic since we can not ensure it being consistently requested as private (or non-private) by all consumers.

Once an interest for non-privately-produced $\mathcal{C}'$ is marked as non-private, $\mathcal{C}'$ must be treated as non-private as long as it remains in R's cache. The rationale for this is as follows: suppose that once $\mathcal{C}'$ has been requested as private, all subsequent requests are delayed. Adv requests $\mathcal{C}'$ twice without privacy. If $\mathcal{C}'$ has been requested before with privacy, *both* requests from Adv will be delayed. On the other hand, if $\mathcal{C}'$ has never been requested before, Adv will experience a cache miss and a cache hit. This clearly allows Adv to determine when $\mathcal{C}'$ has been requested by U *with privacy*.

It is easy to see that, letting the first non-private interest act as a *trigger* (after which interests for corresponding content are not delayed), probing R for $\mathcal{C}'$ does not yield any useful information.

Allowing consumers to set a privacy flag in interests might encourage selfishness: if cached content has been marked as non-private before, they will receive it with no delay. If it has not, they will keep their requests private. The general outcome would be detrimental for all consumers, who would experience high latency even when the requested content is in cache. However, we argue that consumers have at least one incentive for requesting content without privacy: *reduced delay* for re-transmitted interests in case of packet loss.[7] Requesting content with privacy precludes its re-transmission from router caches (if the original content object is lost) and hence results in higher delays. For this reason, we believe that the rational choice for consumers is to request content with privacy only when actually needed.

So far we have considered only techniques where cache hits for private content are always delayed. It is easy to see that, therefore, the approaches listed above are secure according to Definition IV.2, i.e., perfectly private. This is a very security strong notion, and in practice it may not be required. We argue that there are factors – such as content popularity – that allow us to avoid hiding cache hits for private content without significantly compromising privacy. Incidentally, the overall benefits of caches are more evident with popular content. In the next section, we discuss and analyze more sophisticated techniques that take advantage of this observation to offer cache privacy while lowering delay.

[7]Packet loss rate in today's Internet hovers around 4% [3], [2].

VI. Improved Privacy-Utility Trade-Off

Although approaches presented in the previous section provide perfect privacy in the sense of Definition IV.2, they can result in large number of cache misses and long delays experienced by consumers. We now consider more practical techniques that relax the *perfect privacy* requirement in favor of better performance, i.e., higher utility. In general, such techniques randomly decide whether to mimic a cache hit or a cache miss for each content request. The distribution of observed output reflects the (local) popularity of the requested content.

A Non-Private Naïve Approach. Let $c_{\mathcal{C}}$ denote the number of requests for a particular content $\mathcal{C}$. The algorithm always generates a cache miss, iff $c_{\mathcal{C}} \leq k$, where k denotes the size of the anonymity set. Therefore, a cache hit indicates that at least k requests have been generated for $\mathcal{C}$.

This approach is not private, since Adv can determine whether $\mathcal{C}$ was previously requested. To do so, Adv issues requests for $\mathcal{C}$, until it determines (knowing k) that the content is coming from R's cache. Let $c'_{\mathcal{C}}$ be the number of such requests. If $c'_{\mathcal{C}} > 0$, Adv learns that exactly $k - c'_{\mathcal{C}}$ requests have been issued for $\mathcal{C}$.

Random-Cache. Security of the previous scheme depends on Adv's knowledge of k. Our next scheme – *Random-Cache* – selects a random k for each content. Thus, the index of the first cache hit in the output sequence is expected to be random, and should not leak information about the router's cache.

As shown in Algorithm 1, the scheme works as follows: the router maintains a counter $c_{\mathcal{C}}$ for each $\mathcal{C}$. The first request for $\mathcal{C}$ always is a cache miss, and $c_{\mathcal{C}}$ is initialized to 0. Also, $k_{\mathcal{C}}$ is picked from $[0, K)$ according to a distribution on domain $[0, K)$, described by a random variable $\mathcal{K}$. Upon receipt of a new request for $\mathcal{C}$, the router increments $c_{\mathcal{C}}$ and checks whether $c_{\mathcal{C}} \leq k_{\mathcal{C}}$. If so, it generates a cache miss and a cache hit otherwise.

We define *utility* as the ratio of expected number of cache hits and the total number of requests for a given content, i.e., it represents the fraction of interests satisfied from the cache.

Definition VI.1 (Utility)**.** *Let $\mathcal{H}(c)$ denote the random variable describing the distribution of the number of cache hits depending on the total number of requests c ($c \geq 1$). The utility function $u : \mathbb{N} \to \mathbb{R}^+$ of a cache management scheme is defined as: $u(c) = \frac{1}{c}\mathbb{E}(\mathcal{H}(c))$.*

In practice, we derive u using the average number of cache misses, instead of cache hits, which is easier to compute. Let $\mathcal{M}(c)$ denote the random variable describing the distribution of the number of cache misses, based on the total number of requests c ($c \geq 1$). Then, $\mathcal{M}(c) + \mathcal{H}(c) = c$, and $u(c) = 1 - \frac{1}{c}\mathbb{E}(\mathcal{M}(c))$.

Algorithm 1: Random-Caching

1: **Input:** Content $\mathcal{C}$, Domain size K, Distribution of $\mathcal{K}$
2: **Output:** Cache hit or cache miss
3: $\mathbb{T}$:= set of received content
4: **if** $\mathcal{C} \notin \mathbb{T}$ **then**
5: Select $k_{\mathcal{C}}$ from $[0, K)$ with probability $\mathsf{Pr}(\mathcal{K} = k_{\mathcal{C}})$
6: $\mathbb{T} := \mathbb{T} \cup \{\mathcal{C}\}$
7: $c_{\mathcal{C}} := 0$
8: Output cache miss
9: **else**
10: $c_{\mathcal{C}} := c_{\mathcal{C}} + 1$
11: **if** $c_{\mathcal{C}} \leq k_{\mathcal{C}}$ **then**
12: Output cache miss
13: **else**
14: Output cache hit

Specifically, for *Random-Cache*, we have:

$$\mathbb{E}(\mathcal{M}(c)) = \begin{cases} \sum_{i=1}^{c} i \cdot \mathsf{Pr}(\mathcal{K} = i-1) + \sum_{i=c+1}^{K} c \cdot \mathsf{Pr}(\mathcal{K} = i-1), \\ \qquad \text{if } 1 \leq c < K \\ \mathbb{E}(\mathcal{K}), \text{ otherwise.} \end{cases} \tag{1}$$

The distribution $\mathcal{K}$ influences both privacy and utility: If cache misses occur with overwhelming probability, we obtain (almost) perfect privacy with nearly no utility. If $\mathcal{K}$ is uniform, we obtain the best privacy among all distributions in terms of ε (which is 0). In addition, as shown below, we can decrease δ (and improve privacy) by increasing K at the cost of degrading utility. We refer to this instantiation of Random-Cache as *Uniform-Random-Cache*.

Uniform-Random-Cache Let $\mathcal{U}(0, K)$ denote the discrete uniform random variable, i.e., $\mathsf{Pr}(\mathcal{U}(0, K) = r) = 1/K$, $0 \leq r < K$. *Uniform-Random-Cache* is an instantiation of Random-Cache (Algorithm 1) with $\mathcal{K} = \mathcal{U}(0, K)$.

Theorem VI.1 (Privacy)**.** *If all cached content is statistically independent, Uniform-Random-Cache is* $(k, 0, \frac{2k}{K})$*-private.*

Proof: (Sketch) Slightly abusing the notation, let $Q_0(\mathcal{C}, r)$ and $Q_1(\mathcal{C}, r)$ denote the output of Algorithm 1 in states S_0 and S_1, respectively, with $\mathcal{C}$ when $k_{\mathcal{C}} = r$. (Recall that $S_0(\mathcal{C}) = 0$ and $S_1(\mathcal{C}) = x$, where $1 \leq x \leq k$). In addition, $Q_0^t(\mathcal{C}, r)$ and $Q_1^t(\mathcal{C}, r)$ denote the sequence of outputs obtained by executing Algorithm 1 with $\mathcal{C}$ consecutively t times, in states S_0 and S_1, respectively.[8] Let $\mathcal{Q}_0^t$ and $\mathcal{Q}_1^t$ denote two random variables describing $Q_0^t(\mathcal{C}, r)$ and $Q_1^t(\mathcal{C}, r)$ when r is selected uniformly at random according to Line 5 of Algorithm 1.

We show that, for all content $\mathcal{C}$, $\mathcal{Q}_0^t$ and $\mathcal{Q}_1^t$ are $(0, \frac{2x}{K})$-prob. indistinguishable. This implies that *Uniform-Random-Cache* is also $(0, \frac{2x}{K})$-prob. indistinguishable with any $\mathcal{C}$, if all content is statistically independent.

The output of $\mathcal{Q}_0^t$ and $\mathcal{Q}_1^t$ is a sequence with length t consisting of two sub-sequences; the prefix which is composed of consecutive cache misses (i.e., sequence of 0's) and the suffix with consecutive cache hits (i.e., a sequence of 1's).

We partition output space $\Omega = \mathit{Range}(\mathcal{Q}_0^t) \cup \mathit{Range}(\mathcal{Q}_1^t)$ into Ω_1, Ω_2 and Ω_3, for all t and $\mathcal{C}$, as follows:

- $\Omega_1 = \mathit{Range}(\mathcal{Q}_1^t) \setminus \mathit{Range}(\mathcal{Q}_0^t)$: If $r \in [0, x)$, then all the t replies are cache hits in state S_1. However, this output can not appear with S_0 where the very first answer is always a cache miss (the router first needs to retrieve the content). Thus, $\nexists r'$ such that $Q_1^t(\mathcal{C}, r) = Q_0^t(\mathcal{C}, r')$.
- $\Omega_2 = \mathit{Range}(\mathcal{Q}_0^t) \cap \mathit{Range}(\mathcal{Q}_1^t)$: If $r \in [x, K - x)$, then $Q_1^t(\mathcal{C}, r) = Q_0^t(\mathcal{C}, r - x)$.
- $\Omega_3 = \mathit{Range}(\mathcal{Q}_0^t) \setminus \mathit{Range}(\mathcal{Q}_1^t)$: If $r \in [K - x, K)$, then the output with S_0 contains at least $K - x + 1$ cache misses, which is not possible with S_1. Hence, $\nexists r'$ such that $Q_0^t(\mathcal{C}, r) = Q_1^t(\mathcal{C}, r')$.

For output $O \in \Omega$, let $\mathit{prefix}(O)$ denote prefix length of O (i.e., # cache misses in O). Since $k_{\mathcal{C}}$ is selected uniformly at random, for all $O \in \Omega_2$, $\Pr(\mathcal{Q}_0^t = O) = \Pr(k_{\mathcal{C}} = \mathit{prefix}(O) - 1) = \Pr(k_{\mathcal{C}} = \mathit{prefix}(O) + x - 1) = \Pr(\mathcal{Q}_1^t = O)$. Hence, $\varepsilon = 0$. Moreover, if $O \in \Omega_1 \cup \Omega_3$, $\Pr(\mathcal{Q}_0^t = O) + \Pr(\mathcal{Q}_1^t = O) = \frac{1}{K}$. Since $|\Omega_1 \cup \Omega_3| = 2x$, we obtain $\delta = \Pr(\mathcal{Q}_0^t \in \Omega_1 \cup \Omega_3) + \Pr(\mathcal{Q}_1^t \in \Omega_1 \cup \Omega_3) = \frac{2x}{K} \leq \frac{2k}{K}$. ■

This theorem claims that the probability that Adv can determine whether a content has been requested zero or k times is $2k/K$. This is because observing any outcome (hit/miss) which can occur in state S_0, but not in S_1, (or vice-versa) occurs with probability $2x/K$. The analysis also shows that perfect privacy cannot be achieved if a cache hit can be generated, with non-zero probability.

Theorem VI.2 (Utility)**.** *For Uniform-Random-Cache,* $u(c) = 1 - \frac{1}{c}\mathbb{E}(\mathcal{M}(c))$*, where*

$$\mathbb{E}(\mathcal{M}(c)) = \begin{cases} c\left(1 - \frac{c+1}{2K}\right), \textit{ if } 1 \leq c < K, \\ K/2, \textit{ otherwise}. \end{cases}$$

The theorem follows from Equation (1).

Theorems VI.2 and VI.1 show that, by increasing the size of domain K, resulting privacy increases at the cost of degraded utility.

Exponential-Random-Cache. One drawback of uniform distribution is that having only one parameter (K) gives limited flexibility for adjusting the privacy/utility trade-off. Hence, we also consider truncated geometric distribution as a candidate for $\mathcal{K}$. Besides K, the shape of this truncated geometric distribution can be calibrated through an extra parameter. Assigning exponentially larger probability to small values of $k_{\mathcal{C}}$ results in fewer cache misses on average, at the cost of additional privacy loss (ε will increase). The corresponding scheme is called *Exponential-Random-Cache*.

Consider a random variable $\mathcal{G}(\alpha)$ with geometric distribution, i.e., $\mathsf{Pr}(\mathcal{G}(\alpha) = k) = (1 - \alpha) \cdot \alpha^k$, where $k \geq 0$ and $0 < \alpha \leq 1$. Its truncated counterpart denoted by $\widetilde{\mathcal{G}}(\alpha, x_1, x_2)$ has a conditional probability distribution and it is defined as:

[8] Since all content is statistically independent, since: (1) it does not matter whether S_0 and S_1 differ in more than one content's count, and (2) Adv's best strategy is to request the same content multiple times in order to infer information about router state.

$P(\widetilde{\mathcal{G}}(\alpha, x_1, x_2) = k) = \frac{\mathsf{Pr}(\mathcal{G}(\alpha)=k)}{\sum_{i=x_1}^{x_2} \mathsf{Pr}(\mathcal{G}(\alpha)=i)}$ if $x_1 \leq k \leq x_2$ and 0 otherwise, where $[x_1, x_2]$ is the truncation interval ($x_1, x_2 \in \mathbb{N}^0$). Therefore:

$$\mathsf{Pr}(\widetilde{\mathcal{G}}(\alpha, 0, K) = r) = \frac{(1-\alpha)\cdot\alpha^r}{1-\alpha^{K+1}}.$$

Exponential-Random-Cache is an instantiation of *Random-Cache* (Algorithm 1) with $\mathcal{K} = \widetilde{\mathcal{G}}(\alpha, 0, K-1)$. Here, α and K are input parameters of the algorithm that can be calibrated to achieve the desired privacy/utility trade-off:

Theorem VI.3 (Privacy)**.** *If all cached content is statistically independent, Exponential-Random-Cache is* $(k, -k\ln(\alpha), \frac{1-\alpha^k+\alpha^{K-k}-\alpha^K}{1-\alpha^K})$*-private*

Proof: (Sketch) The proof is similar to that of Theorem VI.1. We outline only the main differences. First, we identically partition Ω into Ω_1, Ω_2, Ω_3. If $O \in \Omega_2$,

$$\frac{\Pr(\mathcal{Q}_0^t = O)}{\Pr(\mathcal{Q}_1^t = O)} = \frac{\Pr(k_{\mathcal{C}} = \mathit{prefix}(O) - 1)}{\Pr(k_{\mathcal{C}} = \mathit{prefix}(O) + x - 1)} =$$

$$= \frac{\Pr(\widetilde{\mathcal{G}}(\alpha, 0, K-1) = \mathit{prefix}(O) - 1)}{\Pr(\widetilde{\mathcal{G}}(\alpha, 0, K-1) = \mathit{prefix}(O) + x - 1)} = \alpha^{-x}.$$

Similarly, $\frac{\Pr(\mathcal{Q}_1^t=O)}{\Pr(\mathcal{Q}_0^t=O)} = \alpha^x$. Since $\alpha < 1$, we have $\varepsilon \leq \ln \alpha^{-k}$. In addition, $\Pr(\mathcal{Q}_1^t \in \Omega_1) = \sum_{i=1}^{x} \frac{(1-\alpha)\alpha^{i-1}}{1-\alpha^K} = \frac{1-\alpha^x}{1-\alpha^K}$, and $\Pr(\mathcal{Q}_0^t \in \Omega_3) = \sum_{i=K-x+1}^{K} \frac{(1-\alpha)\alpha^{i-1}}{1-\alpha^K} = \frac{\alpha^{K-x}-\alpha^K}{1-\alpha^K}$. Since $\Pr(\mathcal{Q}_0^t \in \Omega_1) = \Pr(\mathcal{Q}_1^t \in \Omega_3) = 0$, we get $\Pr(\mathcal{Q}_0^t \in \Omega_1 \cup \Omega_3) + \Pr(\mathcal{Q}_1^t \in \Omega_1 \cup \Omega_3) \leq \frac{1-\alpha^k+\alpha^{K-k}-\alpha^K}{1-\alpha^K}$. ∎

Theorem VI.4 (Utility)**.** *For Exponential-Random-Cache,* $u(c) = 1 - \frac{1}{c}\mathbb{E}(\mathcal{M}(c))$*, where*

$$\mathbb{E}(\mathcal{M}(c)) = \begin{cases} \frac{1-\alpha^c-c\alpha^K}{1-\alpha^K} + \frac{\alpha(1-\alpha^c)}{(1-\alpha^K)(1-\alpha)}, \textit{ if } 1 \leq c < K, \\ \frac{1-(K+1)\alpha^K}{1-\alpha^K} + \frac{\alpha}{1-\alpha}, \textit{ otherwise.} \end{cases}$$

Proof: Using the fact that $\sum_{i=1}^{c} i\alpha^{i-1} = \frac{d}{d\alpha}\sum_{i=1}^{c}\alpha^i = \frac{1-(c+1)\alpha^c}{1-\alpha} + \frac{\alpha(1-\alpha^c)}{(1-\alpha)^2}$ and $\sum_{i=c+1}^{K}\alpha^{i-1} = \frac{\alpha^c-\alpha^K}{1-\alpha}$, the theorem follows from Equation (1) after calculation. ∎

Comparison of Proposed Schemes. Increasing α in the Exponential-Random-Cache scheme results in better privacy (smaller ε). However, δ cannot be made arbitrarily small and it is ultimately determined by α. In particular, $\delta = 1 - \alpha^k$ when $K = \infty$, which is the smallest possible δ. In contrast, δ of the uniform distribution can be arbitrarily decreased by sufficiently increasing K.

We compare the utility of our schemes in Figure 4. In Figure 4(a), we adjust the same value of δ, that is 0.05, for both schemes, and plot their utility for different values of k while varying ε. In Figure 4(b), we compute the maximum value of $\varepsilon = -\ln(1-\delta)$ for various combinations of δ and k, and plot the difference between the utility functions of the two schemes for varying δ. Both figures show that the exponential scheme exhibits up to 12% performance gain over the uniform one; Figure 4(a) also shows that both schemes achieve better utility as the number of requests grows.

Addressing Content Correlation. Random-Cache requires statistically independent content in the cache – a very strong assumption. Several content may share the same name prefix, and their access patterns could be strongly correlated. Under this assumption, Random-Cache as described above becomes insecure since it allows Adv to sample multiple points under different k. By requesting a large number of related content, as soon as Adv receives one without any delay it learns that – with overwhelming probability – the whole set of content has been requested before.

To alleviate this problem, correlated content must be grouped together, e.g., by considering elements from the same namespace as a single group. Algorithm 1 can then be applied to these groups rather than to individual content, i.e., using a single counter $c_{\mathcal{C}}$ and value of $k_{\mathcal{C}}$).

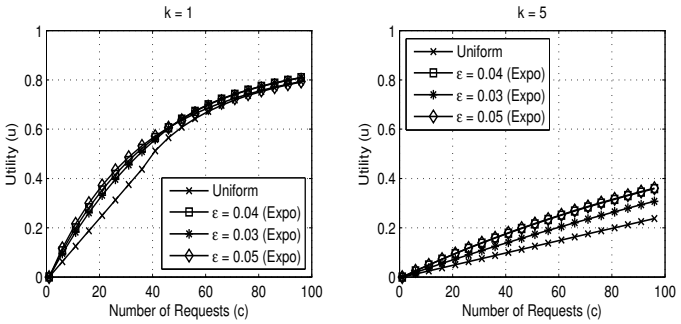
Even the above extension cannot be proven secure against all correlation-based attacks. In many cases, content correlation is even more subtle (e.g., semantically related content having different names such as linked webpages). This might be identified with appropriate background knowledge. As a possible countermeasure, NDN content could be augmented with a content id field. Producers would populate such field with identical values for correlated content. Routers could then determine how to handle such content by observing this field. However, a thorough analysis of these attacks and of the corresponding countermeasures is beyond the scope of this paper.

VII. Experimental Evaluation

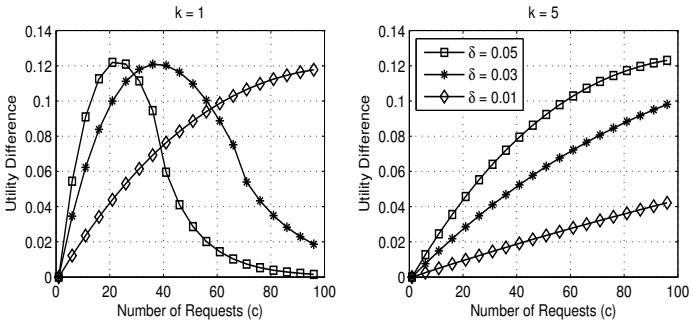
We now evaluate actual impact of our cache privacy techniques through simulations. To do so, we tested them using HTTP traffic traces collected by the IRCache [14] that is part of the The National Laboratory for Advanced Network Research (NLANR) project [22]. Traces were collected on September 1, 2007 (over a 24-hour period) on a Web proxy located at Research Triangle Park, North Carolina. These traces reflect activity of 185 users, for a total of approximately 3.2 million requests distributed over various destinations. We randomly divide requested content into private and non-private. We "re-played" these traces with the following algorithms:

1) **No Privacy.** The router does not use any privacy-preserving technique.
2) **Always Delay Private Content.** For each request of a (cached) *private* content, the router always generates a cache miss, while for *non-private* cached content the response is always cache hit. This implements the basic protocol in Section V-B.
3) **Uniform-Random-Cache/Exponential-Random-Cache.** Requests for cached *private* content are handled according to Algorithm 1. Requests for *non-private* cached content always result in a cache hit.

A router caches all content and removes elements from its cache (when full) according to the LRU policy. In case of



(a) Utility depending on privacy ($\delta = 0.05$)



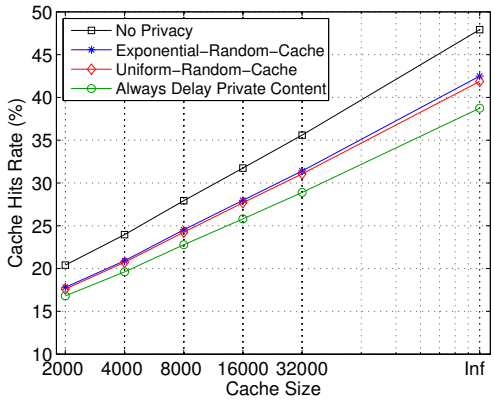
(b) Maximal utility difference between *Uniform-Random-Cache* and *Exponential-Random-Cache* when $\varepsilon = -\ln(1-\delta)$

Figure 4. *Uniform-Random-Cache* vs. *Exponential-Random-Cache*

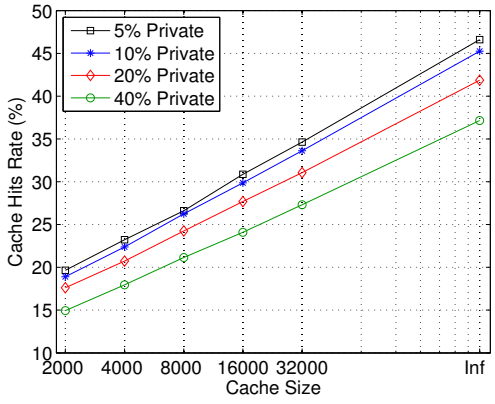
a cache hit, the corresponding cache entry becomes "fresh" even if the response is delayed.

For the algorithms that classify content into private and non-private, each incoming request is randomly marked as private with probabilities 0.05, 0.1, 0.2, and 0.4. Without loss of generality, we assume that all content has the same size. We consider 5 cache sizes: 2,000, 4,000, 8,000, 16,000, and 32,000 content. Furthermore, as a baseline, we also run the same algorithms with an "unlimited" cache.

We set $k = 5$ and $\varepsilon = 0.005$. Results are reported in Figure 5.



(a) Comparison of our techniques



(b) *Exponential-Random-Cache* varying number of private requests

Figure 5. Cache Hit Rates: Experimental Evaluation Results

VIII. Related Work

There is a large body of work on using side channels to extract information about other users' (or applications') behavior.

In [11] and [12], the authors introduce a technique that allows malicious website to learn whether its victim visited a specific web page. The attacker sends a Java applet to the victim, detects cache hits with respect to the user's browsing cache.

Similarly, Felten et al. [10] show how a malicious website can determine whether a web page has been downloaded by its victim. The attack uses a Java applet or Javascript code, and uses timing information to determine the content of the browser's cache. The authors demonstrate the feasibility of the attack through experiments.

Weinberg et. al [25] show interactive techniques that allow an adversary to learn the cache content of users' web browsers, even if basic defense mechanisms are in place.

Baron [1] proposes a countermeasure for the attacks in [11], [12], based on completely hiding browsing history: the rendering behavior of the browser (e.g., link colors, output of CSS functions) with respect to previously visited web pages is identical to that with new pages. However, Baron technique does not work for interactive attacks in [25]. In particular, Weinberg et. al conducted several experiments to show that interactive attacks and also side channel, e.g. timing attacks, can still be exploited by malicious activities to disclose user's previous visited sites.

In [4], Bortz et al. show two types of timing attacks that allow the adversary to learn the content of the browser's cache. The first attack, called *direct timing attack*, reveals whether one or more public website have been visited by the victim. The second, dubbed *cross-site timing attack*, is more dangerous as it can reveal information about private sections of websites; for instance, it is possible to determine whether a user is logged in to specific service.

Another application of side channel attacks is the timing attack on SIP VoIP networks. In [26], the authors provided a tool that can be used to reveal the "calling history" of a SIP domain by observing which "recipient digital certificates" are stored in the local cache. The authors were able to disclose all phone calls between different SIP domains.

Several countermeasure for cache sniffing have been developed. In [16], the authors propose a server-side solution that will protect the users from leaking the content of their cache. The idea is to randomize and personalize the links in web pages so a malicious site cannot guess them when it tries to discover whether they have been visited before.

Schinzel discusses three techniques for mitigating timing-based side channel attacks in web applications [24]. The first technique delays *all* responses such that the total delay of each response is identical. While this approach does not leak any information, it introduces considerable delay and decreases the user experience. The second approach consists in adding a random delay to *each* response (the responses to identical requests referring to the same content are independently randomized). However, requesting the *same* content enough number of times, the adversary can remove this random noise. The third approach is similar to the second one but instead of randomizing the delay per response, a single random delay is selected per destination in order to prevent the aforementioned attack. All these approaches randomize response delay of identical requests independently, which is inefficient in NDN.

Lauinger [18] considers several NDN-related security issues, identifying the problem of cache privacy and overviewing several countermeasures, including some approaches similar to those discussed in this paper. Finally, Crosby et al. [8] investigate how network latency deteriorates due to time-based side channels and design filters to reduce the effects of jitter.

IX. Conclusion

This paper explored cache privacy in NDN and identified several important privacy threats. We then introduced some counter-measures. First, we suggested that consumers and producers should indicate which content is privacy-sensitive. Then, we proposed several techniques that provide certain tradeoffs between privacy and latency. We also introduced a formal model that allows us to quantify the degree of privacy offered by various caching algorithms. We believe that proposed techniques are general and may be of interest beyond NDN caching.

References

[1] L. Baron. Preventing attacks on a users history through css: Visited selectors. *http://dbaron.org/mozilla/visited-privacy*, 2010.

[2] R. Birke, M. Mellia, M. Petracca, and D. Ross. Experiences of voip traffic monitoring in a commercial isp. *IJNM*, 20(5), 2010.

[3] M. Borella, D. Swider, S. Uludag, and G. Brewster. Internet packet loss: Measurement and implications for end-to-end QoS. In *ICPP*, 1998.

[4] A. Bortz and D. Boneh. Exposing private information by timing web applications. In *IW3C*, 2007.

[5] Android Project. http://www.ccnx.org/wiki/working-with-ccnx-android-code/.

[6] CCNx Node Model. http://www.ccnx.org/releases/latest/doc/technical/CCNxProtocol.html.

[7] Content centric networking (CCNx) project. http://www.ccnx.org.

[8] S. Crosby, D. Wallach, and R. Riedi. Opportunities and limits of remote timing attacks. *TISSEC*, 12(3), 2009.

[9] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. Andana: Anonymous named data networking application. In *NDSS*, 2012.

[10] E. Felten and M. Schneider. Timing attacks on web privacy. In *CCS*, 2000.

[11] R. Focardi, R. Gorrieri, R. Lanotte, A. Maggiolo-Schettini, F. Martinelli, S. Tini, and E. Tronci. Formal models of timing attacks on web privacy. *ENTCS*, 62, 2002.

[12] R. Gorrieri, R. Lanotte, A. Maggiolo-Schettini, F. Martinelli, S. Tini, and E. Tronci. Automated analysis of timed security: A aase study on web privacy. *IJIS*, 2(3), 2004.

[13] M. Gotz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Publishing search logs–a comparative study of privacy guarantees. *IEEE TKDE*, 24(3):520 –532, 2012.

[14] IRCache Project. http://www.ircache.net/.

[15] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. In *Co-NEXT*, 2009.

[16] M. Jakobsson and S. Stamm. Web camouflage: Protecting your clients from browser-sniffing attacks. *S&P Magazine*, 5(6), 2007.

[17] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton E. Uzun, B. Zhang, G. Tsudik, K. Claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, E. Yeh. Named Data Networking (NDN) project. Technical report, PARC, 2010.

[18] T. Lauinger. Security & scalability of content-centric networking. Master's thesis, Technische Universitat Darmstadt, 2010.

[19] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, 2008.

[20] Named Data Networking project (NDN). http://named-data.org.

[21] NDN Testbed. http://www.named-data.net/testbed.html.

[22] The National Laboratory for Advanced Network Research Project. http://www.caida.org/projects/nlanr/.

[23] National Science Foundation of future Internet architecture (FIA) program. http://www.nets-fia.net/.

[24] S. Schinzel. An efficient mitigation method for timing side channels on the Web. In *COSADE*, 2011.

[25] Z. Weinberg, E. Chen, P. Jayaraman, and C. Jackson. I still know what you visited last summer: Leaking browsing history via user interaction and side channel attacks. In *Symposium on S&P*, 2011.

[26] G. Zhang, S. Fischer-Hübner, L. Martucci, and S. Ehlert. Revealing the calling history of SIP VoIP systems by timing attacks. In *ARES*, 2009.

[27] X. Zhang, H. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. Andersen. Scion: Scalability, control, and isolation on next-generation networks. In *Symposium on S&P*, 2011.