When Encryption is Not Enough: Privacy Attacks in Content-Centric Networking

Cesar Ghali
Computer Science Department,
University of California Irvine
cghali@uci.edu

Gene Tsudik
Computer Science Department,
University of California Irvine
gene.tsudik@uci.edu

Christopher A. Wood*
Computer Science Department,
University of California Irvine
woodc1@uci.edu

ABSTRACT

Content-Centric Networking (CCN) is a network architecture for transferring named content from producers to consumers upon request. The name-to-content binding is cryptographically enforced with a digital signature generated by the producer. Thus, content integrity and origin authenticity are core features of CCN. In contrast, content confidentiality and privacy are left to the applications. The typically advocated approach for protecting sensitive content is to use encryption, i.e., restrict access to those who have appropriate decryption key(s). Moreover, content is typically encrypted once for identical requests, meaning that many consumers obtain the same encrypted content. From a privacy perspective, this is a step backwards from the "secure channel" approach in today's IP-based Internet, e.g., TLS or IPSec.

In this paper, we assess the privacy pitfalls of this approach, particularly, when the adversary learns some auxiliary information about popularity of certain plaintext content. Merely by observing (or learning) the frequency of requested content, the adversary can learn which encrypted corresponds to which plaintext data. We evaluate this attack using a custom CCN simulator and show that even moderately accurate popularity information suffices for accurate mapping. We also show how the adversary can exploit caches to learn content popularity information. The adversary needs to know the content namespace in order to succeed. Our results show that encryption-based access control is insufficient for privacy in CCN. More extensive counter-measures (such as namespace restrictions and content replication) are needed to mitigate the attack.

CCS CONCEPTS

• **Networks** → **Network architectures**; **Network security**; **Network privacy and anonymity**;

KEYWORDS

content-centric networks; privacy; frequency analysis

*Supported by the NSF Graduate Research Fellowship DGE-1321846.

ICN '17, Berlin, Germany

DOI: 10.1145/3125719.3125723

ACM Reference format:
Cesar Ghali, Gene Tsudik, and Christopher A. Wood. 2017. When Encryption is Not Enough: Privacy Attacks in Content-Centric Networking. In *Proceedings of ICN '17, Berlin, Germany, September 26–28, 2017,* 10 pages.
DOI: 10.1145/3125719.3125723

1 INTRODUCTION

Information-Centric Networking (ICN) is a new networking paradigm that treats content (aka data or information) as a first-class object. Content-Centric Networking (CCN) is a specific type of request-based ICN where a consumer fetches content by issuing an explicit request (called an *interest*) that refers to the desired content by name. The network is responsible for routing an interests towards either a producer of that content or a router that has previously cached it. At every router hop, per-interest state is left behind to allow the content to be sent back, along the same path, thus obviating the need for a "source address" in an interest. Moreover, every router along the way is free to opportunistically cache content in order to satisfy future interests. Subsequent interests that ask for the same content (by the same name) may result in content being served from some intervening router's cache.

From a privacy perspective, CCN suffers from several important issues. One of its key features is that each content must be signed by its producer. In contrast, as a network-layer architecture, CCN does not mandate encryption: content is transferred in cleartext, unless previously encrypted above the network layer. Thus, it is trivial to eavesdrop on names carried in interests and corresponding content payload. If content payload is encrypted, then only the content name is leaked. Ghali et al. [11] recently showed that, in order to minimize this additional leakage, the name contained in an interest must be the output of a keyed deterministic pseudorandom function (PRF) $F_k(\cdot)$. This way, two consumers who request the same content with actual name N must request $F_k(N)$ where k is somehow known to these (presumably authorized) consumers. Eavesdroppers then only learn that two consumers request the same content, and not its actual name.

Ghali et al. also argue in [11] that the above is insufficient from a privacy perspective. In particular, if the adversary has additional auxiliary information about the requested content e.g., its popularity within a given namespace, it can recover the content name even if PRF-transformed names are used. The reason is due to interest linkability, i.e., ability to determine when two interests refer to the same content. We consider it a privacy leakage when the adversary can learn information about underlying interests based on their PRF-transformed names.

This type of leakage is not unique to CCN. If we consider CCN as a generic key-value store where PRF-transformed interests are

keys, and corresponding content packets are values, the problem at hand is analogous to privacy leakage in encrypted databases. This topic has been extensively studied in recent years [27]. In this paper, we apply to CCN attacks from the research literature on privacy of encrypted databases. Specifically, we study adversarial ability to learn plaintexts of requests and responses using only information learned from eavesdropping on encrypted traffic. In doing so, we try to answer the following questions:

- How does the accuracy of adversary's auxiliary information influence effectiveness of privacy attacks?
- How does router caching affect attacks, and how does it relate to topological distribution of the adversary?
- Can replicating or partitioning content among multiple producers decrease effectiveness of these attacks, and if so, to what degree?

This paper makes the following contributions:

- In Section 7, we show that, given accurate auxiliary information about content popularity distribution and sufficiently many request samples, the adversary can, with very high probability, correctly map *some* encrypted content packets to their plaintext counterparts.
- In Section 8, we show that privacy attacks are hindered by router caching and content replication since they effectively reduce the sample size for an adversary.
- In Section 9, we show that knowledge of a namespace is sufficient for the adversary to learn the perceived popularity of content published under that namespace. This has strong implications on how much of a namespace should be public or otherwise discoverable by anyone, especially, when content names correspond to private data.

The rest of the paper is organized as follows. Section 2 overviews CCN. Sections 3 and 4 present our notation and threat model. Next, Section 5 describes the privacy attack and Section 6 discusses the simulator used to study it in various network configurations. Sections 7 and 8 consider the attack in the presence of global and physically distributed eavesdropping adversaries, respectively. Section 9 demonstrates the algorithm used to infer content popularity. Attack implications are discussed Section 10 and related work – in Section 11. The paper concludes with a summary of future work in Section 12.

2 PRELIMINARIES

Content Centric Networking (CCN) is a prominent ICN architecture, originally developed at PARC. Named Data Networking (NDN) [36] is its academic dual. CCN and NDN have minor protocol and packet format differences. In this paper, we focus primarily on the former. This section overviews CCN with respect to the latest specifications [24] and the CCN reference implementation. Given familiarity with either CCN or NDN, it can be skipped without loss of continuity.

In contrast to IP, which focuses on end-points of communication and their names and addresses, CCN [13, 24] focuses on content by making it named, addressable, and routable. A content name is a URI-like [3] string composed of one or more variable-length segments. To obtain content, a user (consumer) issues a request, called an *interest* message, with the name of the desired content. An interest can be *satisfied* by either: (1) a router storing requested content in its cache, or (2) the content producer. In either case, a *content object* message is returned to the consumer. (If a producer can not satisfy an interest, it generates an Interest Return or NACK [7].) Name matching in CCN is exact, e.g., an interest for `/acm/icn/2017/cfp.pdf` can only be satisfied by a content object named `/acm/icn/2017/cfp.pdf`.[1]

Aside from the name, interest messages may include the following optional fields:

- `Payload` – a field that lets consumers push data to producers along with the interest.
- `KeyIdRestriction` – hash of the public key used to verify desired content's signature. If present, CCN guarantees that only content objects that can be verified with the specified key are returned in response to an interest.
- `ContentObjectHashRestriction` – hash of the content being requested. If present, CCN guarantees delivery of content the hash of which matches the value of this field.

Content objects always carry a payload (i.e., the actual content) and some additional metadata. Unlike interests, they also *usually* carry an authenticator, i.e., a signature or a Message Authentication Code (MAC). An authenticator is used to assert correctness of name-to-content binding, and it allows consumers and routers to verify authenticity and integrity of returned content. Content objects do not need to carry a name if the corresponding interest included a `ContentObjectHashRestriction` field. This is because the content can be matched to the interest by computing and checking that its hash equals the corresponding field in the interest. (This check is used to authenticate the response.)

There are three types of entities in CCN:[2] (1) consumer, which issues interests for content, (2) producer, which generates and publishes content, and (3) routers, which forward interest and content messages between consumers and producers. Each CCN entity maintains two components:

- *Forwarding Interest Base* (FIB) – table of name prefixes and corresponding outgoing interfaces. The FIB is used to route interests based on longest-prefix-matching of their names.
- *Pending Interest Table* (PIT) – table of outstanding (pending) interests and, for each, a set of corresponding incoming interfaces.

An entity may also maintain an optional *Content Store* (CS) used for caching content. From here on, we use the terms *CS* and *cache* interchangeably.

A router use its FIB to forward interests towards producers and its PIT – to forward content messages along the reverse path to consumers. More specifically, upon receiving an interest, a router R first checks its cache to see if it can satisfy this interest locally from the cache. When R receives an interest for content named N that is not cached locally and there are no pending interests for the same name in its PIT, R forwards the interest to the next hop according to its FIB. For each forwarded interest, R stores some state information in the PIT, including the name in the interest and the interface on which it arrived, such that content may be sent back to the consumer. If an interest for N arrives while there

[1] In contrast, name matching NDN is longest-prefix-based.
[2] A physical entity, or host, can be both a consumer and producer of content.

is already an entry for the same content name in the PIT, R only needs to update the arriving interface. When content is returned, R forwards it to all of the corresponding incoming interfaces and the PIT entry is removed. If a router receives a content object without a matching PIT entry, the message is silently discarded.

3 NOTATION

Let $\mathcal{D}(\mathbb{U})$ be a probability distribution over some universe of elements $\mathbb{U}$. When it can be inferred from context, we omit $\mathbb{U}$ from $\mathcal{D}(\mathbb{U})$. Let X denote a random variable for a distribution over $\mathbb{U}$. When X is discrete, $f_X(x)$ is the corresponding probability mass function (PMF). For simplicity, we also use $\mathcal{D}(x)$ to denote $f_X(x)$. Given any two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over the same finite domain $\mathbb{U}$, their statistical distance is computed as:

$$\Delta(\mathcal{D}_1, \mathcal{D}_2) \triangleq \sup_{x \in \mathbb{U}} |\mathcal{D}_1(x) - \mathcal{D}_2(x)| = \frac{1}{2} \sum_{x \in \mathbb{U}} |\mathcal{D}_1(x) - \mathcal{D}_2(x)|$$

(This is equivalent to the well-known Kolmogorov-Smirnov statistic.) Frequency distribution $\mathcal{F}_{\mathbb{U}}(n, x)$ represents the number of times $x \in \mathbb{U}$ occurs after taking n samples from $\mathcal{D}(\mathbb{U})$. For any $n > 0$, let r_i denote the i-th largest value of: $\mathcal{F}_{\mathbb{U}}(n, x)|_{x \in \mathbb{U}}$, where $r_1 = \max_{x \in \mathbb{U}}\{\mathcal{F}_{\mathbb{U}}(n, x)\}$ and $r_{|\mathbb{U}|} = \min_{x \in \mathbb{U}}\{\mathcal{F}_{\mathbb{U}}(n, x)\}$.

In CCN, consumers issue requests for content D with name N, denoted as $D(N)$. In this context, N is the *application name* of D. The *network name* $\bar{N}$ carried in the wire-encoded packet and used to forward this request need not be N. In fact, we assume that consumers apply some sort of *obfuscation* or encryption function to N such that $\bar{N} \neq N$. We use $D(N)$ and $D(\bar{N})$ to refer to content identified by the given application and network names, respectively. If $\bar{N}$ is derived from N, then $D(N) = D(\bar{N})$.

After it is requested, $D(N)$ is carried in a content packet $C(\bar{N})$ with the network name $\bar{N}$. Note that different network names $\bar{N}^0$ and $\bar{N}^1$ may be used when requesting $D(N)$, in which case $C(\bar{N}^0) \neq C(\bar{N}^1)$. This can occur if $D(N)$ is uniquely encrypted for different consumers. Conversely, it always holds that: if $C(\bar{N}^0) = C(\bar{N}^1)$, then both responses carry the same application data $D(N)$. Recall that it is not a requirement for a content object to carry a CCN name. However, a content object always carries an explicit or implicit identifier that can be matched to a value computed from the corresponding interest. For example, if $C(\bar{N})$ does not carry a name, then its hash digest must match what is provided in $\bar{N}$. In this case, we use the same notation for the sake of clarity.

4 THREAT MODEL

This section describes the attack on privacy, which is the focus of this paper. We also present the assumed adversary model and auxiliary information.

4.1 Privacy Primer

Recent work on CCN privacy [11] outlined conditions for mitigating a range of eavesdropping attacks. Two measures of privacy were identified: *weak* and *strong*. The former holds if neither a matching interest nor content packet leaks any information except for equality. Strong privacy means that such correlation is impossible; it has strong implications on how interest and content packets are protected in transit. First, both must be protected via semantically secure (or CCA-secure) encryption scheme, which implies that consumers would not benefit from content caching.[3] Moreover, consumers and producers must either share pair-wise keys (for symmetric encryption) or use expensive public-key cryptographic operations to protect packets. Both seem to negate claimed benefits of CCN.

We believe that weak privacy is preferable in order to retain most benefits of CCN. However, it is subject to so-called frequency analysis attacks, if $\mathcal{A}$ has auxiliary information about underlying content [11]. Such information can be extracted from a variety of sources, including the target application (e.g., Netflix or Spotify), publicly-available statistics (e.g., income statistics for a certain demographic or public financial records), or prior versions of content (e.g., a history of the Netflix or Spotify media catalog). In these attacks, $\mathcal{A}$ uses its knowledge about content popularity, along with observed interests, to infer which interest corresponds to which content. As mentioned above, this is similar to attacks on encrypted databases. In that model, $\mathcal{A}$ corrupts a server storing an encrypted database and observes database queries. $\mathcal{A}$'s goal is to determine the plaintext value of each encrypted record based on auxiliary information and observed queries [27].

The attack scenario in the database scenario does not map directly to the frequency analysis attack outlined in [11]. In the former, once $\mathcal{A}$ compromises the target server, it can observe all queries. In contrast, in CCN, $\mathcal{A}$ is a collection of one or more compromised routers that observe network traffic. Therefore, by compromising a single router, $\mathcal{A}$ does not automatically get access to all queries for the target content. This is a critical fact in CCN: distributed nature of the network (particularly, existence of router caches) means that $\mathcal{A}$ has far less information than in the database scenario. In this work, we explore how this gap affects $\mathcal{A}$'s success in attacking CCN privacy.

4.2 Adversarial Model

We assume that $\mathcal{A}$ is a distributed and active adversary that aims to learn information about statically encrypted, or weakly private, content shared among multiple consumers. Encryption is not ephemeral, i.e., packets are not encrypted in transit between producers and consumers. Thus, we assume that $\mathcal{A}$ can correlate interest and (encrypted) content packets referring to the same application data. $\mathcal{A}$ can compromise a subset of routers on the path between arbitrary consumers and the nearest copy of the requested content. Once a router is compromised, $\mathcal{A}$ can observe all packets that it processes. $\mathcal{A}$ can also spawn malicious consumers that probe the network for content and can populate non-compromised routers' caches with copes of that content. However, $\mathcal{A}$ can not compromise either consumers that issue interests for content or producers that respond to interests with encrypted content packets. A realistic example of $\mathcal{A}$ could be a state-sponsored entity or a set of colluding Internet Service Providers (ISPs).

4.3 Adversarial Information

Let $\mathbb{P}$ be a set of application data items, and let $\mathbb{C}$ be the set of encrypted content packets used to carry items in $\mathbb{P}$ through the

[3] Except for the case of interest re-transmission.

network. That is, for each $p \in \mathbb{P}$, there is an encrypted form in $\mathbb{C}$. Let $T : \mathbb{C} \rightarrow \mathbb{P}$ be the *truth mapping* from the encrypted content to their plaintext data counterparts. We assume that all consumers issue interests for content objects $\in \mathbb{P}$ according to some *real popularity distribution* $\mathcal{D}_R(\mathbb{P})$.

$\mathcal{A}$ is given access to some fixed auxiliary information about this popularity distribution, called $\mathcal{D}_A^{\mathcal{A}}(\mathbb{P})$. Moreover, at any time t, $\mathcal{A}$ has access to a *snapshot frequency distribution*, denoted $\mathcal{F}^{\mathcal{A}} : \mathbb{T} \times \mathbb{C} \rightarrow \mathbb{N}$. That is, for each item $c \in \mathbb{C}$, $\mathcal{F}^{\mathcal{A}}(t, c)$ is the number of times c was observed by $\mathcal{A}$ up t. The set of items from $\mathbb{C}$ observed by $\mathcal{A}$ at time t is $\mathbb{O}(t)$. Using $\mathcal{F}^{\mathcal{A}}$, $\mathcal{A}$ can create an *empirical distribution* $\mathcal{D}_E(\mathbb{C})$ of the popularity of each observed item. If $\mathcal{A}$ is a global adversary, then $\mathcal{D}_E(\mathbb{C})$ approximates $\mathcal{D}_T(\mathbb{P})$ under the truth mapping T. Essentially, $\mathcal{D}_A^{\mathcal{A}}(\mathbb{P})$ is $\mathcal{A}$'s approximation of $\mathcal{D}_R(\mathbb{P})$.

5 ATTACK OVERVIEW

We now describe the frequency analysis attack adapted from the encrypted databases scenario [27]. In our setting, encrypted or otherwise obfuscated interests are analogous to queries for encrypted database records. The entire network, comprised of caches and content, is equivalent to one giant database. The adversary can eavesdrop on all (or parts of) the network (database) and, as a result, can view all (or some) interests and content (queries and records). This access, along with auxiliary information about popularity of content (records), is sufficient to perform the attack. More concretely, the core idea is as follows:

> *$\mathcal{A}$ learns (or is given) some auxiliary information about popularity distribution of application names, or content, i.e., $\mathbb{P}$. $\mathcal{A}$ also observes empirical popularity distribution of interests for encrypted content, i.e., $\mathbb{C}$. In doing so, $\mathcal{A}$ seeks to learn T, i.e., which items in $\mathbb{C}$ map to items in $\mathbb{P}$. $\mathcal{A}$ succeeds if it learns any of these with non-negligible success probability.*

In a frequency attack at time t, $\mathcal{A}$ combines $\mathcal{D}_A^{\mathcal{A}}(\mathbb{P})$ and $\mathcal{F}^{\mathcal{A}}(t, c)$, as follows: First, $\mathcal{A}$ ranks items in $\mathcal{F}^{\mathcal{A}}(t, c)$ in order of descending popularity. Then, for each $c_i \in \mathbb{O}(t)$ in decreasing order according to $\mathcal{F}^{\mathcal{A}}(t, c)$, $\mathcal{A}$ guesses that c_i corresponds to the ith most popular item p_j based on $\mathcal{D}_A^{\mathcal{A}}(\mathbb{P})$. Let sort be a function that sorts a histogram in descending order of frequency. Algorithmically, the attack works as follows:

- $\rho \leftarrow \mathrm{sort}(Hist(\mathbb{O}(t)))$
- $\pi \leftarrow \mathrm{sort}(Hist(\mathbb{P}))$
- Compute a mapping $\alpha : \mathbb{C} \rightarrow \mathbb{P}$ such that, for all $c \in \mathbb{C}$:

$$\alpha(c) = \begin{cases} \pi[Rank_\rho(c)] & \text{if } c \in \mathbb{O}(t) \\ \perp & \text{if } c \notin \mathbb{O}(t) \end{cases}$$

The result of the attack is α – the guessed mapping from encrypted data items to their plaintext form.

Accuracy of the attack is defined as follows: Let $R(\alpha, T)$ be a function that counts the number of $\mathcal{A}$'s correct guesses. A correct guess is such that $\alpha(c) = T(c)$. $R(\cdot)$ computes the *total* number of guesses by $\mathcal{A}$. We say the *match percentage* is the total number of correct guesses divided by $|\mathbb{P}|$. By itself, the match percentage may be misleading, e.g., if the dataset is large and has a long tail with items that have near-equal popularities. Thus, we are also interested in *partial accuracy* of the attack. We define partial accuracy as a function $S(\cdot)$ that takes an index $i \leq |\rho|$ along with ρ, α, and T and computes:

$$S(i, \rho, \alpha, T) = \sum_{j=1}^{i} \frac{\sum_{k=1}^{j} \mathrm{Match}(j, \rho, \alpha, T)}{|\rho|},$$

where:

$$\mathrm{Match}(i, \rho, \alpha, T) = \begin{cases} 1 & \text{if } \alpha(\rho[i]) = T(\rho[i]) \\ 0 & \text{if } \alpha(\rho[i]) \neq T(\rho[i]) \end{cases}$$

Intuitively, at index i, this computes percentage of guesses that are correct up to i. For example, it is likely that $S(1, \rho, \alpha, T) \approx 1.0$ if $\mathcal{A}$'s auxiliary information and observances are highly accurate. In some cases, the total accuracy of the attack might not matter, while it might be important if $\mathcal{A}$ can correctly guess only a small number of items with a very high probability.

6 SIMULATING THE ATTACK

We now describe the simulator for evaluating the frequency analysis attack. Its source code is available online at [32].

6.1 Content Distributions

To assess the attack we need realistic information about popularity distributions. Unfortunately, since there are no real-world deployments of CCN (or other ICN architectures), we must rely on information from current web content traces. Fortunately, there has been a great deal of work studying the popularity of web content. Breslau et al. show in [4] that web content does not follow a strict Zipf distribution, as often suggested. Instead, it adheres to a Zipf-like distribution where the ith most popular page is requested with probability proportional to $i^{-\alpha}$, where $\alpha \in [0.6, 2.5]$ [2, 8, 14, 15, 20, 26, 30]. Thus, unless stated otherwise, we hereafter use the Zipf distribution to model real content popularity.

6.2 CCN Simulator

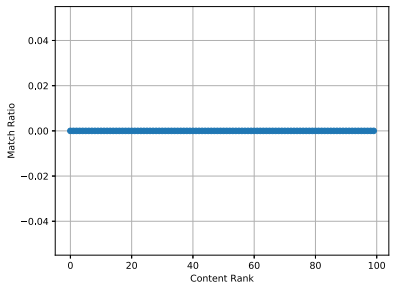
We implemented a custom CCN simulator for this study. We chose not to use available ccns3Sim or ndnSim because we do not need to take into account network behavior at a layer below CCN in the network stack. The attack is sufficiently generic that we only need a way to control interest and contents. Our simulator allows a user to create an arbitrary network topology G composed of sets of: consumers C, routers R, and producer(s) P. Once created, $\mathcal{A}$ is realized as a subset of compromised entities.

Each router has a cache with probability p_c. We represent a network configuration by its topology graph $G = (\mathrm{C}, \mathrm{R}, \mathrm{P}, \mathrm{R}_{\mathrm{Adv}})$, where $\mathrm{R}_{\mathrm{Adv}}$ represents routers controlled by $\mathcal{A}$. A network with cache probability[4] p_c is denoted as $\mathrm{Net}((\mathrm{C}, \mathrm{R}, \mathrm{P}, \mathrm{R}_{\mathrm{Adv}}), p_c)$.

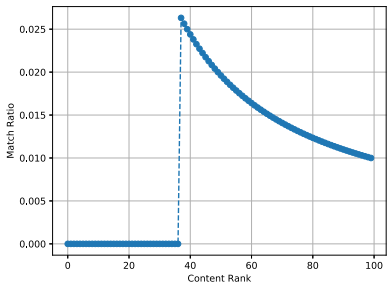
The next step is to create the content universe U. Next, a probability (popularity) distribution is assigned to U, denoted $D(\mathrm{U})$. Consumers sample their interests from this distribution.[5] We also

[4] By cache probability we mean the probability that any node in the network employs a cache. The size of which is bounded to 10, 000 items. This size was chosen so that caches did not suffer constant churn and content was retained for longer than a single consumer-to-producer round trip.

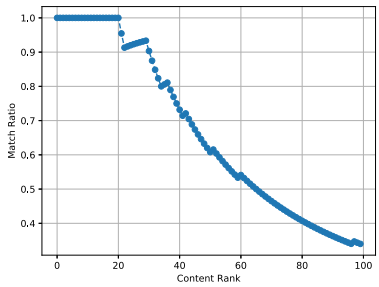
[5] Currently, only Zipf and Uniform distributions are supported. However, it is easy to add, and experiment with, new distributions.



(a) Uniform popularity and auxiliary information



(b) Zipf popularity and uniform auxiliary information



(c) Zipf popularity and auxiliary information

Figure 1: Attack accuracy with varying auxiliary information and content popularity

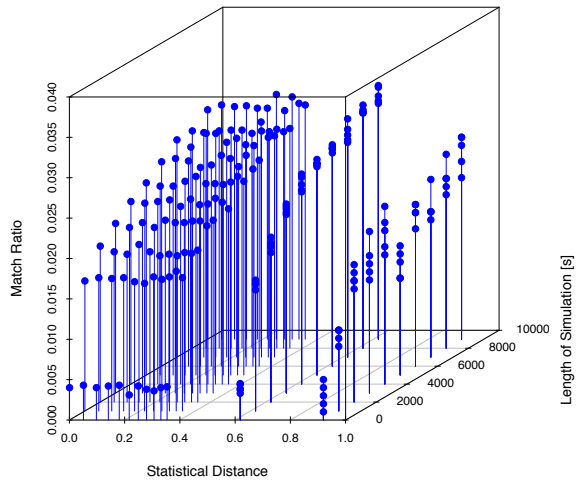


Figure 2: Attack accuracy as a function of $\Delta(\mathcal{D}_R, \mathcal{D}_A)$ and simulation time

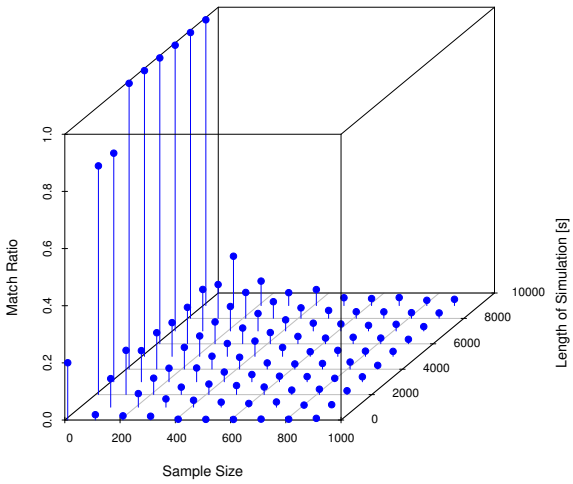


Figure 3: Attack accuracy as a function of content sample size and simulation time

allow auxiliary information distribution $\mathcal{D}_A^{\mathcal{A}}(\mathbb{P})$ to be imposed on the content universe. This distribution is given to $\mathcal{A}$.

Once the simulation is configured, it runs for a number of epochs. At each epoch, a random consumer $c \leftarrow \mathrm{C}$ sends an interest for content $C \leftarrow \mathrm{U}$, i.e., sampled according to true content popularity distribution. An interest is forwarded until it: (1) resuts in a router cache hit, or (2) reaches the producer. Then, a content packet is sent back to the consumer. Each $\mathcal{A}$-controlled node records the interests it sees during this process. When the simulation completes, observed results from each $\mathcal{A}$-controlled node are merged to form $\mathcal{A}$'s complete view of the network. (Specifically, frequency histograms are merged together into one.) This is then fed into the frequency analysis attack along with true popularity distribution and auxiliary information. The output of the attack is the match percentage and $\mathcal{A}$'s accuracy, as described in Section 5

7 GLOBAL EAVESDROPPER ATTACKS

In this section we experimentally assess efficacy of the frequency analysis attack by a global $\mathcal{A}$, denoted by $\mathcal{A}_G$, which is assumed to have access to every interest issued by every consumer in the network. In this model, we need to answer the following question:

> Given content popularity distribution $\mathcal{D}_R$ and $\mathcal{A}_G$ with auxiliary information distribution $\mathcal{D}_A$, to what extent can $\mathcal{A}_G$ successfully correlate encrypted interest and content packets with their plaintext counterparts?

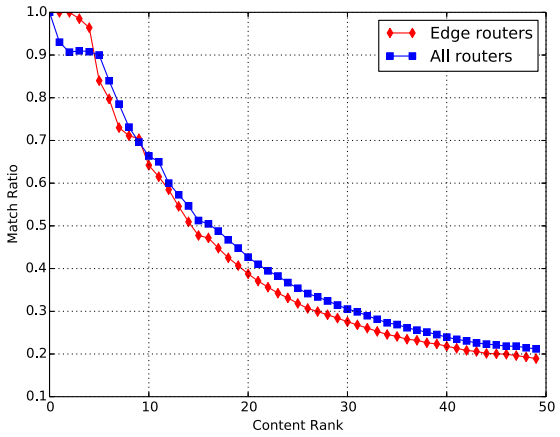


Figure 4: Match percentage for $\mathcal{A}$ distributed across edge and all network routers.

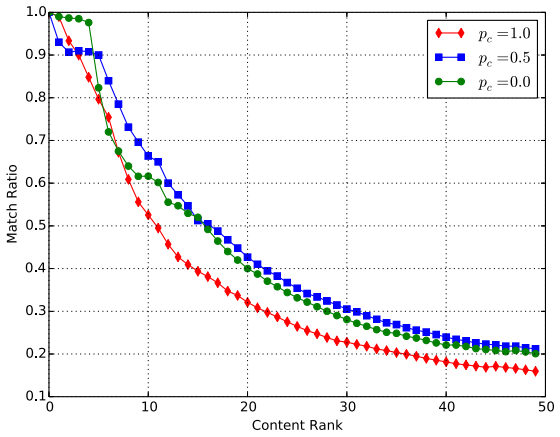


Figure 5: Attack accuracy with varying cache presence in the network

As mentioned in Section 5, we consider total and partial success by $\mathcal{A}_G$, since encryption protects *every* packet equally. We first assess attack accuracy with various $\mathcal{D}_R$ and $\mathcal{D}_A$. Results are shown in Figure 1. With the exception of simulation noise, accuracy is very low when either distribution is uniform. However, when $\mathcal{D}_A$ is statistically close to $\mathcal{D}_R$, attack accuracy becomes very high.

Next, to understand the extent to which statistical distance affects this attack, we conducted the following experiment. First, we created a content universe $\mathbb{U}$ of size N. Then, for each considered probability distribution, we created $\mathcal{D}_R$ and $\mathcal{D}_A$ for $\mathbb{P}$. We considered uniform distribution as a baseline (i.e., the case of $\mathcal{A}_G$ having no auxiliary information) and Zipf distribution with parameters $\alpha \in [0.5, 2.5]$. We then ran the simulator for τ time steps. Finally, we simulated the frequency analysis attack, measured accuracy of resultant guesses, and computed $\Delta(\mathcal{D}_R, \mathcal{D}_A)$. The matching probability, as a function of $\Delta(\mathcal{D}_R, \mathcal{D}_A)$, for various $\mathcal{D}_R$, is shown in Figures 2. It illustrates that, as $\Delta(\mathcal{D}_R, \mathcal{D}_A)$ increases, matching percentage decreases, as expected. However, the rate of decline is low, meaning that even some statistical equivalence is sufficient for the attack.

Size of content universe has a non-negligible effect on attack accuracy. Intuitively, with more options to choose from, $\mathcal{A}_G$'s task of finding the correct mapping becomes more difficult. To show this, we repeated the same experiment as above except with fixed $\mathcal{D}_R$ and $\mathcal{D}_A$, while varying N. Results are shown in Figure 3. As expected, as N increases, matching percentage quickly decreases. This is because each mapping entry becomes more sensitive, as probability space is thinned.

8 DISTRIBUTED EAVESDROPPER ATTACKS

Admittedly, $\mathcal{A}_G$ is not the most realistic adversary. In practice, adversaries will likely be localized in small groups of possibly collocated routers. For example, $\mathcal{A}$ could exploit software running in edge access points to observe traffic closest to consumers, or it could subvert an AS and compromise some or all of its routers.

We now consider a distributed adversary under a variety of scenarios, in order to assess the relationship between network caching, content location, and $\mathcal{A}$'s topological distribution. Each of these variables impacts the type and number of samples observed by $\mathcal{A}$, which are the main components of the attack. As quality of this information degrades, so should attack accuracy.

We conducted all experiments described below over a topology based on Deutsches ForschungsNetz (DFN). It consists of 160 consumers, multiple producers attached to edge routers, and multiple routers (more than 30).

8.1 Adversary Distributions and Caching Effects

Attack accuracy increases as a function of $\mathcal{A}$'s coverage. As shown in the previous section, accuracy can be quite high if $\mathcal{A}$ can observe all traffic. However, as $\mathcal{A}$'s presence declines, so does the number of samples observed. We consider two $\mathcal{A}$ topological configurations: (1) distributed among some fraction of *edge* routers, and (2) distributed among a random fraction of *all* routers. To explore the impact of $\mathcal{A}$'s topological distribution, we conducted an attack experiment on:

$$\mathrm{Net}((\mathrm{C}, \mathrm{R}, \mathrm{P}, \mathrm{R}_{\mathrm{Adv}}), p_c)$$

where $p_c = 0.5$ and $\mathrm{R}_{\mathrm{Adv}}$ is nearly 25% of edge routers or 25% of all routers. Results in Figure 4. show that, in the edge case, $\mathcal{A}$ attains higher accuracy for high ranking content. This is because its knowledge of interest frequency is more complete, due to duplicate interests not being masked by caches.

Caching also plays an important role: if enabled in every router, there should be, in theory, less traffic traversing the network. Thus, $\mathcal{A}$ would *observe fewer* samples of encrypted content[6]; thus attack

[6] Assuming that $\mathcal{A}$ is not located at the edge.

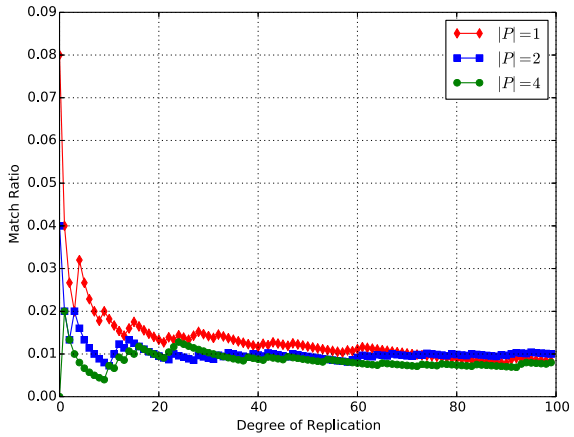


Figure 6: Attack accuracy with producer replication.

accuracy would necessarily decline. This is an interesting relationship explored in [1]. In some scenarios, caching can be easily exploited to violate privacy of individual consumers. However, with respect to content, caching complicates the attack.

To explore this relationship, we experimented with:

$$\mathrm{Net}((\mathrm{C},\mathrm{R},\mathrm{P},\mathrm{R}_{\mathrm{Adv}}),p_c)$$

where $p_c \in \{0.0, 0.5, 1.0\}$. Each router has a 0.25 probability of being compromised. Results in Figure 5 show that, when caching is disabled, $\mathcal{A}$ is correct (for high ranking content) with greater probability than if caching is globally enabled. This is a direct result of observing fewer samples.

8.2 Replication Effects

Replication is another important factor in attack efficacy. Similar to caching, replication allows interest and content packets to bypass $\mathcal{A}$ and cause it to observe less traffic. To understand the extent to which replication deters the frequency analysis attack, we conducted experiments on $\mathrm{Net}((\mathrm{C},\mathrm{R},\mathrm{P},\mathrm{R}_{\mathrm{Adv}}),p_c)$, where $p_c = 0.5$ and $|\mathrm{P}| \in \{1,2,4\}$. A replicated producer publishes the same content under a different prefix[7] in a different part of the network. (We forced each replica to be sufficiently disjoint topologically in order to to reduce the chance of two interests for different replicas traversing the same path.) As shown in Figure 6, attack accuracy decreases as the degree of replication increases. With more replicas, consumers are free to stripe their interests across multiple prefixes and, by doing so, potentially bypass $\mathcal{A}$.

9 POPULARITY INFERENCE

The frequency analysis attack requires $\mathcal{D}_E(\mathbb{C})$ and $\mathcal{D}_A^{\mathcal{A}}(\mathbb{P})$. The former is needed to approximate $\mathcal{D}_R(\mathbb{P})$. The latter is not always readily available by eavesdropping. Fortunately, by exploiting properties of CCN, it can be learned. Armed with knowledge of the public namespace for a set of content, i.e., network names $\bar{N}$, $\mathcal{A}$ (acting as a malicious consumer) can build $\mathcal{D}_A^{\mathcal{A}}(\mathbb{P})$ by probing the network. (Recall that, in this work, network names are encrypted, and the adversary can not learn their contents. Thus, knowledge of the network name does not lead to knowledge of the corresponding application name.) Specifically, $\mathcal{A}$ can query for known names and, based on response time, infer whether corresponding content is cached. This sort of inference attack is similar to the invasive cache probe attack in [18], which works as follows: $\mathcal{A}$ aims to learn whether some nearby consumer asked for content named N. To do so, it requests N and relies on timing information to learn whether this content was served from a cache. (Response time lower than the consumer-to-producer RTT means that requested content was served from a cache.)

The rate at which it probes the network is important since $\mathcal{A}$ must be able to differentiate between cached copies that it itself injected into the network by probing, and actual cached copies previously requested by nearby consumers. Let t_c be the characteristic time of a cache [6]. If LRU cache eviction policy is used, then t_c is average time elapsed between the last request for an item and its eviction. If the cache uses FIFO or random policy, then t_c is average time between insertion and eviction. For simplicity, we assume that all routers use LRU. t_c tells $\mathcal{A}$ the expected time interval before an item is evicted. In turn, this can be used as a lower bound on probe frequency f_p, i.e., $f_p > 1/t_c$.

Assuming $\mathcal{A}$ knows t_c, the probing algorithm runs as follows: Let $\mathrm{AppendRandomComponent}(N,\lambda)$ be a function that, given input name N, samples a random bit-string of length λ and appends it to N as the last name component. This effectively creates a unique name for which a cache hit is impossible, with overwhelming probability. Given name universe $\mathcal{N}$, $\mathcal{A}$ iterates through the set and, for each $N \in \mathcal{N}$, issues a pair of interests (probes): $N_h = N$ and $N_m = \mathrm{AppendRandomComponent}(N, 128)$, and records time t_N. It then waits to receive the corresponding content packets (named N_h and N_m), and records their respective arrival times t_N^h and t_N^m.[8] When both are received, $\mathcal{A}$ computes:

$$\Delta_N = ||(t_N^h - t_N)| - |(t_N^m - t_N)||$$

If $\Delta_N > \epsilon$ for some constant ϵ, $\mathcal{A}$learns that an interest for N_h was satisfied faster than that for N_m, which went all the way to the producer. Therefore, $\mathcal{A}$ learns that a cache hit occurred. $\mathcal{A}$ then sleeps for t_c before proceeding to the next name in $\mathcal{N}$. This process is repeated to incrementally build the EDF of the namespace. This procedure is shown in Algorithm 1, where r is the number of namespace iterations to complete before producing the estimated auxiliary popularity map ρ, and t_c is a known characteristic time.

Before assessing this algorithm, we consider its runtime, which is approximately $r|\mathcal{N}|t_c\mathrm{RTT}_{max}$, where RTT_{max} is the maximum RTT for any interest probe-pair. This is clearly infeasible as $\mathcal{N}$ grows. Therefore, we recommend implementing a parallelized variant of this algorithm. Specifically, instead of traversing $\mathcal{N}$ in sequence, $\mathcal{A}$ can do it in parallel and only sleep for t_c in between each successive probe for N. Given P processing units, runtime is lowered to $rt_c\frac{|N|}{P}\mathrm{RTT}_{max}$, since each name can be processed in parallel.

[7] For example, an Akamai replica might use the `/akamai`, while a Fastly replica might use the `/fastly` prefix.

[8] Recall that Interest Returns are sent in response to interest requesting non-existing content. This guarantees that every request receives a response, barring any packet loss.

Algorithm 1 InferPopularity

Input: $\mathcal{N}, r, t_c, \epsilon$
Output: $\alpha : \mathcal{N} \rightarrow \mathbb{N}$
for $N \in \mathcal{N}$ **do**
  $\alpha[N] = 0$
end for
for $i = 1, \ldots, r$ **do**
  for $N \in \mathcal{N}$ **do**
    $N_h = N; N_m = \text{AppendRandomComponent}(N, 128)$
    $t_N = \text{now}()$
    Send requests for N_h and N_m in parallel and record their time of arrival in t_N^h and t_N^m
    $\Delta_N = ||(t_N^h - t_N)| - |(t_N^m - t_N)||$
    if $\Delta_N > \epsilon$ **then**
      $\rho[N] = \rho[N] + 1$
    end if
    Sleep for t_c
  end for
end for
return α

We evaluate this algorithm as follows. Using ccns3Sim [29], we created a simulation that contained n consumers $Cr_1, \ldots, Cr_n$, one "monitor" Cr^*, ($\mathcal{A}$), and a single producer with S content objects. We chose $S = 50$ and $S = 100$ for our simulations here to illustrate the efficacy of this attack. Each consumer is given access to all these content names. Popularity of content in this collection followed a Zipf distribution with $\alpha = 1.5$. Nodes were arranged in a star topology with a single caching router between them. Storage size of the caching router matched that of the content collection. Each consumer, Cr_i,$i = 1, \ldots, n$, requested a random name from the content collection every second. Meanwhile, Cr^* executed Algorithm 1. At the end of the simulation, *actual* frequency distribution of content (as perceived) by the producer and the *observed* frequency distribution of Cr^* are collected and shown in Figure 7. It demonstrates that the attack algorithm can learn, with fairly high accuracy (for the given values of S), the actual popularity distribution solely by exploiting router caches. (Of course, this accuracy may decline as S increases further. We plan to explore this degradation in future work.)

The popularity inference attack works if the namespace is *static* and enumerable. However, it no longer applies if the namespace is dynamic or otherwise unpredictable.

9.1 Estimating Characteristic Time

Probing frequency f_p must be large enough such that all router caches on the $\mathcal{A}$-to-producer path evict stale content between sequential probes for the same content. Therefore, $\mathcal{A}$ needs to know maximum t_c* for all on-path routers. Assuming interests for N issued by other consumers follow a Poisson Arrival process with rate λ_N, probability that a probe for N results in a cache hit (for any cache) can be characterized as in [9]:

$$h_N = 1 - e^{-\lambda_N t_c*}$$

For a given R_i on the $\mathcal{A}$-to-producer path, t_c^i is a constant that satisfies:

$$\sum_{N \in \mathbb{N}} (1 - e^{-\lambda_N t_c^i}) = C,$$

where C is capacity of the LRU cache. To approximate t_c*, $\mathcal{A}$ must therefore know: (a) capacity of each on-path router and (b) namespace from which content can be requested. Though possible, it is highly unlikely that $\mathcal{A}$ would learn all this information. Therefore, it must be approximated. One trivial way to do this is to assume a large C and modest λ_N to represent each cache and all names. This would not yield a value close to t_c*. However, since the inference algorithm does not require precision in the characteristic time, this is acceptable.

10 DISCUSSION

We now discuss the efficacy of the privacy attack and importance of caching and namespace enumeration in mitigating this attack.

10.1 Attack Efficacy

Success of the attack relies on three key pieces of information: $\mathcal{D}_A^{\mathcal{A}}(\mathbb{P})$, $\mathcal{D}_R(\mathbb{P})$, and $\mathcal{D}_E(\mathbb{C})$. If $\mathcal{D}_A^{\mathcal{A}}(\mathbb{P})$ is *perfect*, i.e., $\Delta(\mathcal{D}_A^{\mathcal{A}}(\mathbb{P}), \mathcal{D}_R(\mathbb{P})) \approx 0.0$, the attack's success depends on accuracy of empirically observed popularity distribution. This is because $\mathcal{A}$ uses its knowledge of observed popularity to map items in $\mathbb{P}$ to $\mathbb{C}$. Put another way, the attack is most successful when the following conditions hold:

$$\Delta(\mathcal{D}_A^{\mathcal{A}}(\mathbb{P}), \mathcal{D}_R(\mathbb{P})) \approx 0.0$$
$$\Delta(\mathcal{D}_E(\mathbb{C}), \mathcal{D}_A^{\mathcal{A}}(\mathbb{P})) \approx 0.0$$

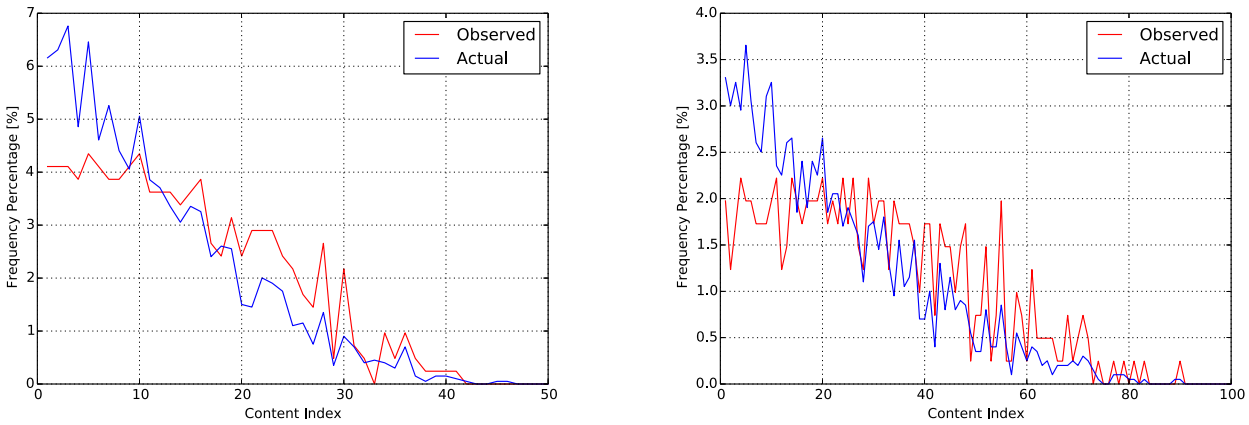
As shown earlier, $\mathcal{A}$'s topological placement, the use of caching, and distribution of content across the network all play a role in widening the gap between these distributions. One definite countermeasure requires forcefully widening the gap between any of these distributions. If $\mathcal{A}$ is localized within a part of the network, then distributing content across multiple locations can help bypass it. However, if $\mathcal{A}$ is topologically wide-spread, enabling caching limits the number of events observed, thus making attacks more difficult. Ultimately, attack's efficacy is a function of $\mathcal{A}$'s location.

10.2 Caching and Privacy

Based on our discussion thus far, it is evident that caching can both help and hurt privacy. Lauinger et al. [18] and Acs et al. [1] showed that a cache can be exploited as an oracle to allow $\mathcal{A}$ to learn when popular content is requested by nearby consumers. This attack uses a timing side-channel based on router caches. It works by requesting popular content from the network via interest "probes" and trying to discern if they have been served from a nearby cache. Similarly, probing attacks that target content producers can be used to discover whether certain content was recently served. One mitigation strategy is to remove the timing channel by requiring caches to artificially delay responding to interests that are marked as "private".

In this paper, we showed that caching can *help* privacy by hindering $\mathcal{A}$'s ability to conduct frequency analysis attacks. These attacks do not *directly* rely on the timing side-channel. Instead, they rely on the content popularity side-channel. The only step that relies on timing is when $\mathcal{A}$ estimates content popularity distributions. Without caches, interest frequency is not dampened before reaching $\mathcal{A}$. Therefore, caching is encouraged. Moreover, timing side-channel mitigation from [1] does not make this attack any easier; it only increases latency for consumers.

An alternate attack mitigation strategy is to protect interest and content packets with semantically secure encryption (see [11] for



(a) $S = 50$ **items** **(b)** $S = 100$ **items**

Figure 7: EDF inference attack accuracy

more details). This would obviate any on-path caching and make each interest-content pair unique. However, though it would mitigate frequency analysis attacks, impact on the network (especially, in terms of congestion) would likely be substantial.

10.3 Namespace Enumeration

If $\mathcal{D}_A^{\mathcal{A}}(\mathbb{P})$ is unknown, feasibility of the frequency analysis attack relies on $\mathcal{A}$'s ability to enumerate the content namespace. This is possible if: $\mathcal{A}$ (a) knows all content names *a priori*, (b) can discover or learn them through external means, (c) can derive them by some namespace convention, e.g., if namespace structure is well-defined, or (d) can learn them from some search engine. Regardless of the method, public content is always enumerable. (If it were not, no one would be able to access it.) Therefore, restricting or controlling enumeration is only possible for restricted content, for which privacy is probably the most important. Thus, if there are access control mechanisms that require consumer authentication before requesting content, enumeration would be restricted to only authorized consumers. This might suffice for some applications. However, consider a CCN-based media distribution service similar to Netflix. Every authorized consumer has access to (essentially) the same content and discovery mechanisms. Since access is pervasive across the entire content collection, enumeration is not preventable. For example, in the Netflix example, every client can search for and discover the same protected content.

Note that shortening the lifetime of a name-to-content binding does not help prevent enumeration. CCN requires every content object to have at least one name. Therefore, a producer could update the name-to-data bindings at regular intervals. However, if $\mathcal{A}$ can discover this binding in a given time epoch, it can also almost surely do the same for the next epoch. What matters for the attack considered in this work is how many times *a specific content* is requested, and not how many times a *specific name* is requested.

11 RELATED WORK

Privacy in CCN and related architectures was first addressed in [5]. They highlight several areas where privacy is at risk and mention possible fixes or mitigations. Ghali et al. [11] formalize the notion of data privacy and show under what circumstances privacy is and is not possible in CCN. So-called weak privacy requires requests and content to be encrypted, as is presumed here. Our results show that in practice this is often insufficient. Inference attacks on encrypted database systems were recently studied by Naveed et al. [27]. Their results inspired our work.

Privacy issues outside of the core request and response protocol have been explored in the context of caches [1, 22, 23]. These results focus on determining whether two consumers requested the same content by "probing" network caches for content based on its name. Strong request privacy deters such attacks while weak request privacy only minimizes their likelihood of success. This type of probing attack was first explored by Lauinger et al. [17–19].

As shown by Ghali et al. in [11], encryption is a necessary though not sufficient condition for privacy in CCN. Many variations of encryption-based access control exist, including those built on group-based encryption [31], broadcast encryption [21], attribute-based encryption [12], and proxy re-encryption [34]. Kurihara et al. [16] and Yu et al. [35] provided ways to generalize access control mechanisms – the former using explicit configuration via CCN manifests and the latter using NDN names. Interest-based access control [10] protects requests instead of responses. (However, it does not preclude responses being encrypted as well.) The main idea is that the content name can only be derived by authorized consumers.

Transparent packet security, i.e., network-layer encryption without application awareness, was proposed by Wood in [33]. A TLS-like key exchange protocol used to build secure sessions for ephemeral, end-to-end security in CCN was described by Mosko et al. in [25]. A network-layer tunnel for similar privacy guarantees was designed and implemented by Oliviera-Nunes et al. in [28].

12 CONCLUSION AND FUTURE WORK

This paper investigated privacy attacks on the CCN architecture that are based on content popularity. As discussed above, $\mathcal{A}$ equipped with some auxiliary information and the ability to eavesdrop on traffic, can correctly map statically encrypted content packets to their plaintext counterparts. This attack does not work with dynamically generated content. Beyond $\mathcal{A}$'s position in the network, this attack is dependent on the degree of router caching and content replication. Lastly, we showed that knowledge of a namespace is sufficient for $\mathcal{A}$ to approximate the popularity of content in that namespace. Results suggest that, in order to mitigate attacks based on this information, content namespaces should be restricted or otherwise not enumerable.

For future work, we plan to verify many of our experimental findings with a companion qualitative analysis. We will also improve and optimize the custom simulator to enable faster experimentation. Moreover, we plan to develop a way for network and adversary topological information to be crafted more easily, thereby enabling more comprehensive experimentation. Lastly, we plan to adapt our frequency analysis attack and apply it on existing IP-based applications.

REFERENCES

[1] G. Acs, M. Conti, P. Gasti, C. Ghali, G. Tsudik, and C. Wood. 2017. Privacy-Aware Caching in Information-Centric Networking. *IEEE Transactions on Dependable and Secure Computing* PP, 99 (2017), 1–1. DOI: https://doi.org/10.1109/TDSC.2017.2679711

[2] Walter Bellante, Rosa Vilardi, and Dario Rossi. 2013. On Netflix catalog dynamics and caching performance. In *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2013 IEEE 18th International Workshop on*. IEEE, 89–93.

[3] Tim Berners-Lee, Roy Fielding, and Larry Masinter. 1998. RFC 2396: Uniform resource identifiers (URI): generic syntax. (1998).

[4] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. 1999. Web caching and Zipf-like distributions: Evidence and implications. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 1. IEEE, 126–134.

[5] Abdelberi Chaabane, Emiliano De Cristofaro, Mohamed Ali Kaafar, and others. 2013. Privacy in content-oriented networking: Threats and countermeasures. *ACM SIGCOMM Computer Communication Review* 43, 3 (2013), 25–33.

[6] Hao Che, Zhijun Wang, and Ye Tung. 2001. Analysis and design of hierarchical web caching systems. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 3. IEEE, 1416–1424.

[7] Alberto Compagno, Mauro Conti, Cesar Ghali, and Gene Tsudik. 2015. To NACK or not to NACK? negative acknowledgments in information-centric networking. In *Computer Communication and Networks (ICCCN), 2015 24th International Conference on*. IEEE, 1–10.

[8] Danny De Vleeschauwer and Koen Laevens. 2009. Performance of caching algorithms for IPTV on-demand services. *IEEE Transactions on broadcasting* 55, 2 (2009), 491–501.

[9] Mostafa Dehghan, Bo Jiang, Ali Dabirmoghaddam, and Don Towsley. 2015. On the analysis of caches with pending interest tables. In *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 69–78.

[10] Cesar Ghali, Marc A Schlosberg, Gene Tsudik, and others. 2015. Interest-based access control for content centric networks. In *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 147–156.

[11] Cesar Ghali, Gene Tsudik, and Christopher A Wood. 2016. (The Futility of) Data Privacy in Content-Centric Networking. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*. ACM, 143–152.

[12] Mihaela Ion, Jianqing Zhang, and Eve M Schooler. 2013. Toward content-centric privacy in ICN: Attribute-based encryption and routing. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*. ACM, 39–40.

[13] Van Jacobson, Diana K Smetters, James D Thornton, and others. 2009. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 1–12.

[14] Chamil Jayasundara, Ampalavanapillai Nirmalathas, Elaine Wong, and Nishaanthan Nadarajah. 2010. Popularity-aware caching algorithm for video-on-demand delivery over broadband access networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 1–5.

[15] Konstantinos Katsaros, George Xylomenos, and George C Polyzos. 2011. MultiCache: An overlay architecture for information-centric networking. *Computer Networks* 55, 4 (2011), 936–947.

[16] Jun Kurihara, Christopher Wood, and Ersin Uzuin. 2015. An Encryption-Based Access Control Framework for Content-Centric Networking. *IFIP Networking* (2015).

[17] Tobias Lauinger. 2010. *Security & scalability of content-centric networking*. Master's thesis. Technische Universität.

[18] Tobias Lauinger, Nikolaos Laoutaris, Pablo Rodriguez, Thorsten Strufe, Ernst Biersack, and Engin Kirda. 2012. Privacy implications of ubiquitous caching in named data networking architectures. *Technical Report TR-iSecLab-0812-001, ISecLab, Tech. Rep.* (2012).

[19] Tobias Lauinger, Nikolaos Laoutaris, Pablo Rodriguez, Thorsten Strufe, Ernst Biersack, and Engin Kirda. 2012. Privacy risks in named data networking: what is the cost of performance? *ACM SIGCOMM Computer Communication Review* 42, 5 (2012), 54–57.

[20] KY Leung, Eric WM Wong, and Kai-Hau Yeung. 2004. Designing efficient and robust caching algorithms for streaming-on-demand services on the Internet. *World Wide Web* 7, 3 (2004), 297–314.

[21] Satyajayant Misra, Reza Tourani, and Nahid Ebrahimi Majd. 2013. Secure Content Delivery in Information-Centric Networks: Design, Implementation, and Analyses. In *ICN*.

[22] Aziz Mohaisen, Hesham Mekky, Xiaobing Zhang, and others. 2015. Timing attacks on access privacy in information centric networks and countermeasures. (2015).

[23] Abedelaziz Mohaisen, Xinwen Zhang, Max Schuchard, Haiyong Xie, and Yongdae Kim. 2013. Protecting access privacy of cached contents in information centric networks. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 173–178.

[24] Marc Mosko, Ignacio Solis, and Christopher A. Wood. 2017. *CCNx Semantics*. Internet-Draft draft-irtf-icnrg-ccnxsemantics-04. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-ccnxsemantics-04 Work in Progress.

[25] Marc Mosko, Ersin Uzun, and Christopher A. Wood. 2017. Mobile Sessions in Content-Centric Networks. *IFIP Networking* (2017).

[26] Luca Muscariello, Giovanna Carofiglio, and Massimo Gallo. 2011. Bandwidth and storage sharing performance in information centric networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. ACM, 26–31.

[27] Muhammad Naveed, Seny Kamara, and Charles V Wright. 2015. Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 644–655.

[28] Ivan Oliviera-Nunes, Gene Tsudik, and Christopher A. Wood. 2017. Namespace Tunnels in Content-Centric Networks. *to appear in 42nd Annual IEEE Conference on Local Computer Networks* (2017).

[29] PARC. 2017. ccns3Sim: CCNx Module for NS3. (2017). https://github.com/parc/ccns3sim

[30] Elisha J Rosensweig, Jim Kurose, and Don Towsley. 2010. Approximate models for general cache networks. In *INFOCOM, 2010 Proceedings IEEE*. IEEE, 1–9.

[31] Diana K. Smetters, Philippe Golle, and J. D. Thornton. 2010. *CCNx Access Control Specifications*. Technical Report. PARC.

[32] Christopher A. Wood. 2017. CCN eavesdropper simulator. (2017). https://github.com/chris-wood/ccn-eavesdropper-simulator

[33] Christopher A. Wood. 2017. Protecting the Long Tail: Transparent Packet Security in Content-Centric Networks. *IFIP Networking* (2017).

[34] Christopher A. Wood and Ersin Uzun. 2014. Flexible End-to-End Content Security in CCN. In *CCNC*.

[35] Yingdi Yu, Alexander Afanasyev, and Lixia Zhang. 2015. Name-based access control. *Technical Report NDN-0034, University of California, Los Angeles, Los Angeles* (2015).

[36] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, and others. 2014. Named data networking. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 66–73.