(The Futility of) Data Privacy in Content-Centric Networking

Cesar Ghali   Gene Tsudik   Christopher A. Wood[*]
Computer Science Department, University of California Irvine
{cghali, gene.tsudik, woodc1}@uci.edu

ABSTRACT

Content-centric networking is an architecture designed to transfer named and addressable data from producers to consumers. Data retrieval is driven by a simple request and response protocol. A consumer issues a request for named data that is routed by the network towards the nearest location where this data is stored. Once found, the corresponding data is returned to the consumer. This data-centric model is different from the datagram- and stream-based protocols used to transport data between endpoints in IP networks: Instead of being tied to the channel through which data flows, security and privacy properties apply to data itself. Consequently, privacy issues in CCN warrant careful evaluation. In this paper, we present a comprehensive assessment of CCN privacy issues in the presence of various adversaries. We specify conditions sufficient to achieve different levels of privacy. We also show that data privacy is more dependent on requests than responses for data. We conclude that strong privacy necessitates some form of session- or channel-based communication, which strongly contradicts the data-centric nature of CCN. We also discuss how to implement proposed CCN privacy mechanisms in practice.

1. INTRODUCTION

Named data is the focal point of Content-Centric Networking (CCN) and related Information-Centric Networking (ICN) architectures. Instead of transferring data between two endpoints identified by IP addresses, a CCN endpoint (or consumer) obtains data by issuing a request (interest) for content[1] using its name. The network is responsible for using the name to route this request to the nearest location that can serve the content. Once the content is located, either at the producer or in a router cache, it is forwarded back to the consumer along the reverse-path of the corresponding interest.

This communication model has subtle yet important implications on privacy. Forwarders (routers) and producers match requests to responses via *exact name matching*. All interests must carry, at a minimum, a name that is used for routing *and matching*. The implication is that if two consumers wish to fetch the same content cached in a router, they must issue *identical interests* for that content. Thus, an eavesdropper can easily correlate multiple requests for the same content. Without going further, privacy is already compromised by the use of equality-based matching and presence of shared router caches.

At present, the baseline for privacy in the Internet is that all traffic must be encrypted in a forward-secure, end-to-end fashion, e.g., as in TLS [22]. Therefore, correlating information across multiple flows and their packets is infeasible (modulo traffic and timing analysis attacks). Clearly, CCN and related ICN architectures do not offer this degree of privacy. To the best of our knowledge, this disparity has not been adequately addressed by the ICN research community. To this end, this paper analyzes CCN *data* privacy issues and shows how requests, responses, and a single request-response exchange can be made private. The contributions are:

- Privacy analysis of the CCN request and response protocol.
- Evaluation of weak privacy techniques based solely on CCN names and their secrecy. This can be used to bootstrap stronger privacy communication techniques.
- Discussion of practical techniques that can address identified privacy issues.

One of our primary conclusions is that data encryption, by itself, is insufficient for privacy. Request names must have no correlation with data carried in a response, which strongly contradicts the name-based model of CCN and other ICNs. This implies that there must be some mapping between standard CCN names and information conveyed to the network. Moreover, in the presence of powerful adversaries, this transformation cannot be deterministic since that would lead to frequency analysis attacks. This effectively invalidates caches and is functionally no different from end-to-end encryption such as TLS in today's IP-based networks. In total, we find that, if privacy similar to IP is desired, then many of the claimed benefits of CCN are lost, barring major architectural changes made to accommodate enhanced levels of privacy.

The rest of this paper is organized as follows. Section 2 gives an overview of CCN. Section 3 outlines the concept of data privacy and presents the threat model. Section 4 assesses what privacy is possible in the presence of an eavesdropping adversary. Section 5 focuses on the same problems with a significantly stronger adversary. Section 6 shows that weaker forms of privacy can fail in practice given auxiliary information about content popularity. We then discuss how to implement different levels of privacy in Section 7. Finally, we conclude with related work in Section 8 and future work in Section 9.

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

[1]We use the terms content and data synonymously.

WPES'16, October 24 2016, Vienna, Austria
© 2016 ACM. ISBN 978-1-4503-4569-9/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2994620.2994639

2. CCN OVERVIEW

Content Centric Networking (CCN) is one of the main ICN architectures. Named Data Networking (NDN) [28] is its academic dual with minor protocol and packet format differences. Therefore, we focus primarily on CCN in the remainder of this paper. This section overviews CCN with respect to the latest specifications [19] and the CCNx 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 [12, 19] 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. This interest can be *satisfied* by either (1) a router storing the content in its cache or (2) the content producer. A *content object* message is returned to the consumer upon satisfaction of the interest. Name matching in CCN is exact, e.g., an interest for `/facebook/Alice/profile.html` can only be satisfied by a content object named `/facebook/Alice/profile.html`.[2]

Aside from the `Name` field, interest messages may include the following fields:

- `Payload` – a field that lets consumers push data to producers along with the request.
- `KeyIdRestriction` – an optional hash digest of the public key used to verify the desired content's digital signature. If present, the network guarantees that only content objects which can be verified with the specified key will be returned in response to an interest.
- `ContentObjectHashRestriction` – an optional hash value of the content being requested. If this field exists, the network guarantees the delivery of the exact content that consumer requests.

Together, all of these fields make up the general "name" of the interest message. Similar to interests, content objects also carry a payload and some additional metadata. However, unlike interests, they also usually carry an authenticator (digital signature or MAC) used to assert the correctness of the name-to-data binding. This authenticator allows consumers and routers to verify the authenticity and integrity of content. Content objects do not need to carry a name if the corresponding interest carried a `ContentObjectHashRestriction` field. This is because the content can be matched to the interest by computing and checking its hash for equality with the corresponding field in the interest. (This equality check is also used to authenticate the response.)

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

- *Forwarding Interest Base* (FIB) – a 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) – a table of outstanding (pending) interests and a set of corresponding incoming interfaces.

An entity may also maintain an optional *Content Store* (CS) used for caching. The timeout for cached content is specified in the `ExpiryTime` field of the content header. From here on, we use the terms *CS* and *cache* interchangeably.

[2]Name matching is long-prefix-based in NDN.

[3]A physical entity, or host, can be both a consumer and producer of content.

Routers use the FIB to forward interests towards producers and the PIT to forward content object 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 of the interest and the interface from which it arrived, so that content may be sent back to the consumer. If an interest for N arrives while there 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. DATA PRIVACY IN CCN

According to [21], "the common definition of privacy in the cryptographic community limits the information that is leaked by the distributed computation to be the information that can be learned from the designated output of the computation." In networking and communication protocols, this notion of privacy refers to the limits of information leaked by traffic. This is a growing concern in recent years since pervasive monitoring of network traffic was found to be standard practice. Such eavesdropping is now considered a fundamental attack on privacy [8]. Other threats to privacy include correlation among a user's traffic flows or behaviors, identification of the user, disclosure of their personal information, and secondary-use of user information [6].

With the shift from host-based to data-centric communication, CCN changes how data is retrieved and the way peers communicate. Recall that consumers issue a request for data D with the name N, which we denote 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 equal to N. However, in standard CCN, $N = \bar{N}$. We use the notation $D(N)$ and $D(\bar{N})$ to refer to the data identified by the given application and network names, respectively. These requests may contain other information to identify the desired content object, such as `KeyIdRestriction` or `ContentObjectHashRestriction`. In this paper, we consider these and all other "identifiers" to be contained in N and subsequently encoded in $\bar{N}$.

After it is requested, $D(N)$ is carried in a content object message $C(\bar{N})$ with the network name $\bar{N}$. Note that consumers may use different network names $\bar{N}^0$ and $\bar{N}^1$ 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 each consumer. 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 the content object $C(\bar{N})$ 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 presentation clarity.

Privacy in CCN must be defined and assessed with respect to the requests and responses that are conveyed in the network, i.e., $\bar{N}$ and $C(\bar{N})$. Generally, an adversary Adv may try to recover N or $D(N)$ from this information. Our goal in the remainder of this paper is to show what type of privacy is attainable based on the properties of $\bar{N}$ and $C(\bar{N})$. Before doing so, we specify our notion of data privacy and how it differs from its IP counterpart. We then describe the adversarial model and define the privacy terms used in the remainder of the paper.

3.1 Separating Privacy and Anonymity

In general, privacy is framed in terms of the endpoints that participate in a data exchange rather than a property of the data itself. For example, privacy might mean that the communicated data leaks no private information about the user. Another notion of privacy might be that identities of communicating endpoints (e.g., IP addresses, host names, user-IDs) remain hidden. We claim that elements of data privacy and personal privacy, or anonymity, are inappropriately mixed in general discussions of privacy. Therefore, we separate these two notions and focus solely on problems surrounding data privacy.

To show why this separation is useful, consider the following scenario. Suppose an adversary controls a pair of consumer- and producer-adjacent routers. Any (unmodified) requests and responses forwarded through the compromised routers link the consumer and producer. The adversary learns that (a) the consumer is fetching data from the producer or (b) the consumer and producer are communicating. However, **we do not consider this a data privacy leak** because consumer and producer linkability does reveal information about data transported between them.

Assuming appropriate protection of requests and responses (as discussed in the remainder of this paper), consumer and producer linkability reveals no more than what is revealed by clients and servers engaged in a TLS session in today's Internet. Although the consumer and producer do not have anonymity, their traffic remains private. If anonymity is required, a TOR-like mechanism such as ANDANA [7] or AC3N [24] can be used to decouple consumers from producers. In the rest of this paper we focus solely on the problem of data privacy.

3.2 Adversarial Model

We now define the adversarial model against which privacy will be assessed. According to [6], there are at least three types of adversarial goals:

1. **Correlate**: Determine whenever any two consumers retrieve the same application data $D(N)$.
2. **Identify**: Discover or recognize that $D(N)$ was retrieved.
3. **Learn**: Obtain information about $D(N)$.

Learning information about $D(N)$ from $\bar{N}$ or $C(\bar{N})$ is harder than a correlation or identification attack. If an adversary learns information about $D(N)$ given $C(\bar{N})$ or $\bar{N}$, then it can also perform a correlation or identification attack. The converse is not necessarily true. We denote the adversaries with these goals as Adv^C, Adv^I, and Adv^L, respectively.

An adversary with any subset of these goals may have different capabilities. One type might only be able to observe a single request and response from a consumer, while another type might observe all traffic from a set of consumers. One example of the latter would be a malicious Wi-Fi hot spot observing traffic of all hot spot users. Adversaries are also classified based on whether they are off-path or on-path, i.e., honest-but-curious (HbC) routers. An adversary that can only capture traffic without being on the consumer-to-producer path has a distinct disadvantage compared to the one that forwards traffic between a consumer and producer. For instance, an off-path adversary eavesdropping on an encrypted link can only observe encrypted traffic. Conversely, routers incident to that encrypted link can observe the original packets. In that case, the adversary can correlate requests with responses more easily and can also determine when two downstream consumers request identical or related content. A more powerful adversary controls multiple routers on the consumer to producer path. For example, an adversary which controls routers adjacent to communicating consumers and producers can easily learn when a particular consumer is requesting a specific content by simply inspecting the name.

Table 1: Adversary examples

	Goals	
Capabilities	**Correlate**	**Extract**
Eavesdropper	A user spying on encrypted traffic between a hotspot and neighbors with the goal of identifying traffic patterns and commonalities.	A user spying on encrypted traffic to identify specially marked or flagged content.
On-path HbC	An access point gathering statistics about how frequently content is accessed.	An access point censoring or restricting content based on its name or payload.
Distributed on-path	A pair of access points adjacent to consumers and a single producer trying to discern when certain content is requested and by whom.	A pair of access points logging when specific content is requested.

We consider these capabilities to be characteristic of three distinct adversaries: an off-path attacker, a single on-path honest-but-curious router, and a collection of at least two (distributed) on-path honest-but-curious routers. Table 1 shows examples of these adversaries.

3.3 Response Privacy

Response privacy is about trying to prevent information leakage from data responses. Ideally, an adversary should learn nothing from a response. We use a game-based definition to capture this notion of privacy. Let $\mathbb{N}_A$ and $\mathbb{N}_N$ be the set of application and network names, respectively, that can be assigned to data and bound to content packets. For every request for $D(N)$, $N \in \mathbb{N}_A$, there is a function $r(N)$ that generates and returns a response $C(\bar{N})$, where $\bar{N} \in \mathbb{N}_N$ (i.e., a network name for $D(N)$). In this game, Adv is given access to $r(\cdot)$ via an oracle $\mathcal{O}_r(\cdot)$. Upon receipt of a name N, $\mathcal{O}_r(N)$ returns $C(\bar{N})$. Adv also has access to an oracle to compute the inverse of $r(\cdot)$, $\mathcal{O}_r^{-1}(\cdot)$, which, upon receipt of $C(\bar{N})$ will return $N = r^{-1}(C(\bar{N}))$. The game works as follows.

- The challenger initializes and gives Adv access to $\mathcal{O}_r(\cdot)$, $\mathcal{O}_r^{-1}(\cdot)$, and $\mathbb{N}_A$.
- Adv issues a series of application names $N_0, \ldots, N_{i-1}$ to $\mathcal{O}_r(\cdot)$ and obtains $C(\bar{N})_0, \ldots, C(\bar{N})_{i-1}$, respectively.
- Adv generates two names $N_i^0 \neq N_i^1$ and sends them to the challenger. The challenger then generates a random bit b and returns $C(\bar{N})^b = r(N_i^b)$ and sends it back to Adv.
- Adv continues to query $\mathcal{O}_r(\cdot)$ (including, if needed, names N_i^0 and N_i^1). Adv can also query $\mathcal{O}_r^{-1}(\cdot)$ for any data response (except $C(\bar{N})^b$). When done, Adv outputs a single bit b'.
- The game outputs 1 if $b = b'$ and 0 otherwise.

We denote the output as $\mathsf{DataGame}(\mathsf{Adv}, r)$. Adv wins if $\mathsf{DataGame}(\mathsf{Adv}, r) = 1$. We also consider one other variant of this game where Adv cannot access either oracle, denoted as $\mathsf{LIMDataGame}(\mathsf{Adv}, r)$.

We define privacy with respect to the function $r(\cdot)$, e.g., data is private with respect to correlation attacks if it is generated by a function $r(\cdot)$ that is also secure against correlation attacks.

Definition 1. *$r(\cdot)$ is secure against correlation attacks if for any probabilistic polynomial time (PPT)* Adv *it holds that*

$$|\Pr[\mathsf{DataGame}(\mathsf{Adv}, r) = 1] - \Pr[\mathsf{DataGame}(\mathsf{Adv}, r) = 0]| \leq \epsilon(\lambda)$$

for security parameter λ and where $\mathsf{Adv} = \mathsf{Adv}^C$.

Similarly, $r(\cdot)$ is secure against leakage and identification attacks if for any PPT Adv *it holds that*

$$|\Pr[\mathsf{LIMDataGame}(\mathsf{Adv}, r) = 1] - \Pr[\mathsf{LIMDataGame}(\mathsf{Adv}, r) = 0]| \leq \epsilon(\lambda)$$

for security parameter λ where $\mathsf{Adv} \in \{\mathsf{Adv}^L, \mathsf{Adv}^I\}$.

We now define weak and strong response privacy with respect to correlation and leakage privacy.

Definition 2. *A response has* weak privacy *if it is generated by a function secure against leakage and identification attacks. A response has* strong privacy *if it has weak privacy and is generated by a function secure against correlation attacks.*

3.4 Request Privacy

Similar to response privacy, request privacy is about information leaked by a network name. However, the contents of a response may compromise the privacy of the corresponding request. This is possible if $C(\bar{N})$ reveals information about N or $D(N)$, e.g., if $C(\bar{N})$ is part of a well-known media file. This complicates our notion of request privacy since we must also assume Adv can observe a response associated with each request. Specifically, let Adv be an adversary similar to Adv^I whose goal is to determine $D(N)$ given $\bar{N}$ (i.e., to recognize that name $\bar{N}$ corresponds to some data item $D(N)$). We assume that application names are transformed by some function $q(\cdot,\cdot)$ to a network representation before interests are issued. We model this transformation as $q : \mathbb{N}_A \times \mathbb{N}^+ \to \mathbb{N}_N$, where $\mathbb{N}_N$ is the set of network names. (We require $|\mathbb{N}_A| \leq |\mathbb{N}_N|$.) The second parameter (from $\mathbb{N}^+$) denotes the length of the application name prefix that *is not modified by the transformation*. For example, $q(\texttt{/a/b/c}, 1)$ would translate the suffix `/b/c` but leave the prefix `/a/` intact. In the current CCN model, $q(\cdot,\cdot)$ is simply the identity function: an application name N is the same name that would be carried by an interest in the network. Clearly, this is problematic for privacy.

We now define a request indistinguishability game. In it, Adv is given access to $q(\cdot,\cdot)$ via an oracle $\mathcal{O}_{q,i^*}(\cdot)$ that computes the transformation after segment i^* of the name. That is, given a name N, $\mathcal{O}_{q,i^*}$ returns $\bar{N} = q(N, i^*)$. Adv also has access to an oracle to compute the inverse of $q(\cdot,\cdot)$, $\mathcal{O}^{-1}_{q,i^*}(\cdot)$, which, upon receipt of $\bar{N}$ returns $N = q^{-1}(\bar{N}, i^*)$. The game proceeds as follows.

- The challenger initializes and gives Adv access to $\mathcal{O}_{g,i^*}(\cdot)$ and $\mathcal{O}^{-1}_{g,i^*}(\cdot)$. Adv is also given $\mathbb{N}_A$ and $\mathbb{N}_N$.
- Adv issues a series of names $N_0, \ldots, N_{j-1}$ to $\mathcal{O}_{q,i^*}(\cdot)$ and collects the results $\bar{N}_0, \ldots, \bar{N}_{j-1}$.
- Adv presents a pair of names N^0_j and N^1_j to the challenger. It is required that the first i segments of N^0_j and N^1_j are identical. The challenger generates a random bit b and returns $\bar{N}^b_j = q(N^b_j, i^*)$ to Adv.
- Adv continues to make queries to $\mathcal{O}_{q,i^*}(\cdot)$. Adv can also issue a request for any name as well as query $\mathcal{O}^{-1}_{q,i^*}(\cdot)$ with transformed names except $\bar{N}^b_j$. Next, Adv outputs a single bit b'.
- The game outputs 1 if $b' = b$ and 0 otherwise.

We denote the output as $\mathsf{NameGame}(\mathsf{Adv}, q)$. Adv succeeds if $\mathsf{NameGame}(\mathsf{Adv}, q) = 1$. We also consider a variant of this game where Adv has no access to the oracles. In this case, Adv relies solely on the response from the challenger and its queries to the network when making its decision. We refer to this as the $\mathsf{INDNameGame}$.[4] We use it to define name privacy with respect to $q(\cdot,\cdot)$.

Definition 3. *$q(\cdot,\cdot)$ is secure against correlation attacks if for any PPT* Adv *it holds that*

$$|\Pr[\mathsf{NameGame}(\mathsf{Adv}, q) = 1] - \Pr[\mathsf{NameGame}(\mathsf{Adv}, q) = 0]| \leq \epsilon(\lambda)$$

for security parameter λ where $\mathsf{Adv} = \mathsf{Adv}^C$.

Similarly, $q(\cdot,\cdot)$ is secure against leakage and identification attacks if for any PPT Adv *it holds that*

$$|\Pr[\mathsf{INDNameGame}(\mathsf{Adv}, q) = 1] - \Pr[\mathsf{INDNameGame}(\mathsf{Adv}, q) = 0]| \leq \epsilon(\lambda)$$

for security parameter λ where $\mathsf{Adv} \in \{\mathsf{Adv}^I, \mathsf{Adv}^L\}$.

We now define weak and strong request privacy.

Definition 4. *A request has* weak privacy *if it is secure against leakage and identification attacks. A request has* strong privacy *if it has weak privacy and is secure against correlation attacks.*

3.5 Communication Privacy

If a request and response are both private, then the entire exchange is private. As before, there is both a strong and weak form of communication privacy. We define both with respect to request and response privacy as follows.

Definition 5. *A request and response have* weak communication privacy *if* **at least** *the request and response have weak privacy. Similarly, a request and response have* strong communication privacy *if* **both** *the request and response have strong privacy.*

4. PRIVACY AGAINST EAVESDROPPERS

Eavesdropping is the weakest attack on privacy. An eavesdropping adversary might not be able to capture any packet at will and may not be able to observe all request and response pairs. It can, however, collect some traffic for offline analysis. In this section we show how this simple capability has strong implications on protection mechanisms for mitigating the main privacy threats outlined in Section 3.

4.1 Data Generation Functions and Response Privacy

Privacy against leakage and correlation attacks for responses is closely related to the concepts of indistinguishable and chosen-ciphertext-attack (CCA) security [13]. We show this in Theorems 1 and 2.

Theorem 1. *Responses must be encrypted with IND-secure encryption to have weak privacy.*

Proof. We define $r(\cdot)$ in $\mathsf{LIMDataGame}$ to be an IND-secure encryption function over generated data. To show that responses are secure against leakage attacks, we must show that Adv only

[4]This variant corresponds to Adv that does not possess offline computation resources.

Algorithm 1 SuffixHash

```
Input: $N, i, F(\cdot)$
Output: $\bar{N}$
1: $\bar{N} = N[1:i], l = |N|$
2: for $j = i+1 \to l$ do
3:     $\bar{N}[j] = F(N[j])$
4: end for
5: return $\bar{N}$
```

Algorithm 2 SuffixHashFlatten

```
Input: $N, i, F(\cdot)$
Output: $\bar{N}$
1: $\bar{N} = \phi, l = |N|$
2: for $j = 1 \to i$ do
3:     $\bar{N}$.Append$(N[j])$
4: end for
5: $\bar{N}$.Append$(F(N[i+1]||\ldots||N[l])$
6: return $\bar{N}$
```

wins the LIMDataGame with probability negligible in λ. In this case, Adv generates and submits N^0 and N^1 to the challenger which generates a random bit b, and returns $D^b = r(N^b)$, as before. Now, consider an alternate IND-secure encryption function $h(\cdot)$ that encrypts N^0 or N^1 instead of data to which these names are bound. Since there is a one-to-one correspondence between names and data (i.e., N^0 and N^1 are bound to D^0 and D^1, respectively), encrypting D^0 or D^1, using $h(\cdot)$, is no different from directly encrypting N^0 or N^1, using $r(\cdot)$. Therefore, Adv's probability of successfully guessing b by examining D^b is no more than its probability of correctly guessing b by examining encrypted N^0 and N^1. The latter is bounded by the probability of Adv correctly guessing b in the IND-secure encryption game, which is negligible. □

THEOREM 2. *Responses must be encrypted with CCA-secure encryption to have strong privacy.*

PROOF. We define $r(\cdot)$ in DataGame to be a CCA-secure encryption function over generated data. As in the proof of Theorem 1, names form a one-to-one correspondence with data. Thus, there is no difference between encrypting names or corresponding data using $r(\cdot)$. Also, if $r(\cdot)$ were to encrypt names, then DataGame would be identical to the standard CCA-secure encryption game. Therefore, since the probability of Adv winning such a game is negligible in λ, it follows that Adv's probability of winning DataGame is also negligible in λ. □

4.2 Name Transformations & Request Privacy

Information leaked from the request gives Adv some advantage in winning NameGame and INDNameGame. Therefore, to assess the capabilities of Adv we must capture information leaked by $q(\cdot,\cdot)$. To begin, observe that the vanilla CCN identity function $q(\cdot,\cdot)$ provides no privacy. Since the identity function does not change names in any way. Adv's probability of correctly matching the challenge name to one of its inputs is always 1.0.

Clearly, the identity function is a trivial transformation, so we explore alternatives. Transformation functions may operate on one or more name segments at a time. For example, consider the translation function in Algorithm 1. F is a cryptographic hash function, which transforms each segment after a given prefix segment number into its hash digest. We call this type of transformation is *structure preserving* since it does not change the hierarchy of name segments; it only changes each name segment value. A transformation is not structure preserving if it modifies hierarchical information in a name. For example, the transformation in Algorithm 2 is not structure preserving. This is because it replaces the suffix after index i with the hash of that suffix. Thus, any names that have more than $l > i$ segments will always be transformed into a name with exactly $(i+1)$ segments.

A transformation is *uniform* if, for every input, it produces output of the same length. This means that uniform structure preserving transformations transform each individual segment of a name to some fixed-length value. Conversely, a uniform non-structure-preserving transformation yields suffixes of equal length.

We now consider some guiding criteria for achieving different levels of request privacy.

THEOREM 3. *Weak and strong request privacy transformations are uniform and non-structure-preserving.*

PROOF. Assume that $q(\cdot,\cdot)$ is a transformation that is neither uniform nor structure-preserving. However, it enables weak and strong request privacy. Let N^0 and N^1 be two Adv-chosen names with different numbers of segments. Upon receipt of N^0 and N^1 in NameGame, the challenger returns $\bar{N}^b = q(N^b, i^*)$. Since $|N^0| \neq |N^1|$, it is trivial for Adv to determine b since either $|\bar{N}^b| = |N^0|$ or $|\bar{N}^b| = |N^1|$. This contradicts the assumption that $q(\cdot,\cdot)$ provides strong and weak privacy.

Now consider N^0 and N^1 that have the same number of name segments but, for at least one segment i, $|N^0[i]| \neq |N^1[i]|$. Upon receipt of N^0 and N^1 in NameGame, the challenger returns $\bar{N}^b = q(N^b, i^*)$. To determine b, Adv looks for the segment i where $|\bar{N}^b[i]| = |N^0[i]|$ and $|\bar{N}^b[i]| \neq |N^1[i]|$ (or vice versa). This exists since $q(\cdot,\cdot)$ is not a uniform transformation. Therefore, $q(\cdot,\cdot)$ does not provide strong or weak privacy since Adv can always win NameGame. □

We conclude that structure-preserving transformations offer neither weak nor strong request privacy. Taking this into consideration, there are at least two types of cryptographic primitives we can use to build transformations suitable for request privacy: hash and encryption functions. Hash functions are uniform by definition. However, encryption functions are not necessarily uniform. Therefore, encryption-based transformations must involve some form of padding to ensure uniformity.

Consider the SuffixHashFlatten transformation in Algorithm 2. This function replaces the suffix of the input name with its hash. In practice, however, using F does not yield request privacy. Since F is a publicly computable function, the unpredictability (entropy) of its output is directly related to the entropy of its input.

Let $x \in X$ denote an element in the support of a random variable X. Traditional Shannon entropy H of X is defined as

$$H(X) = -\sum_{x \in X} P(X = x)\log(P(X = x)).$$

The chain rule is useful to quantify entropy lost when new information is presented. Formally, the entropy of X, if conditioned on the entropy of Y, can be decreased by at most the latter, i.e., $H(X|Y) = H(X,Y) - H(Y)$, where $H(X,Y)$ is the joint entropy of X and Y defined as

$$H(X,Y) = -\sum_{x \in X}\sum_{y \in Y} \Pr(X = x, Y = y)\log(\Pr(X = x, Y = y)).$$

This formulation can be generalized to support computing the conditional and joint entropy of an arbitrary number of random variables. With it, we can quantify the entropy of names if we treat individual name segments as Discrete Random Variables (DRVs) and an entire name as a sequence of DRVs. For example, names of k segments (or components) can be viewed as a sequence of k DRVs. By using a large sample set of names, we can compute the entropy of the i-th segment DRV as well as its conditional entropy based on all $j < i$ segment DRVs. As input data we used the Cisco URI dataset available from [1]. It consists of $13,549,122$ unique URIs. The average length of each URI is 57.4B with median 52B and standard deviation 33.182B. Across all URIs, the average number of segments is 6.67 with median length 6 and standard deviation 2.212 segments. Using this dataset, compute the entropy of individual name segments. The results, which are plotted in Figure 1, show that single-segment entropy is skewed with the peak at the 5th name segment. When name segments are considered in unison the results are quite different. Figure 2 shows how conditional entropy significantly degrades as more name segments (components) are considered. The implication is that the prefix of a name leaks a significant amount of information about its suffix. While this dataset may not be representative, we believe that results would be similar for larger and even more diverse datasets.

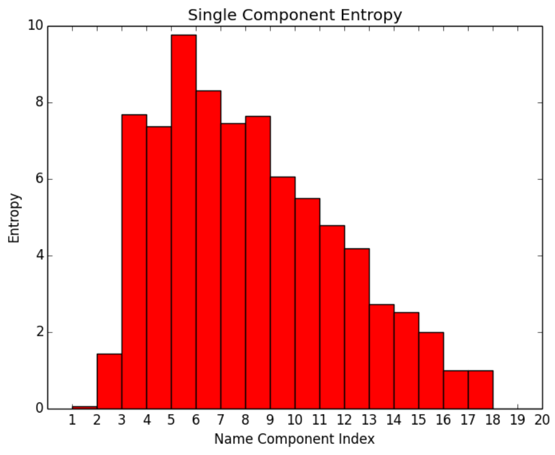

Figure 1: Single component entropy

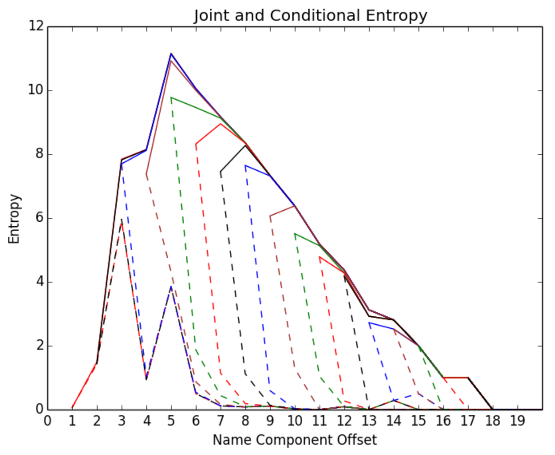

Figure 2: Joint and conditional across multiple segment entropy

These results lead us to conclude that (parts of) content names have very little entropy and are highly predictable. This is an intuitive result since URIs are meant to be meaningful and therefore predictable. Thus, we need strong randomness guarantees for the name translation function. It must not leak any information about the input application names. Consequently, it must be at least an IND-secure encryption function. Or, more generally, it must be a cryptographic pseudorandom function (PRF). PRFs are derived from PRF families that take additional inputs, such as a random seed or key to produce a specific PRF. Someone with knowledge of this additional input can compute the output of the PRF for any input since these functions are deterministic. For weak request privacy, the PRF cannot be computed by Adv. Otherwise, Adv could compute the PRF value without the oracle and easily win NameGame. This is captured in the following theorem.

THEOREM 4. *A name transformation function must be a PRF to enable weak request privacy.*

PROOF. The proof follows from the discussion above. If the transformation is a PRF then Adv cannot compute the transformation output without the oracle. Moreover, by the properties of the PRF, the outputs are independent and uniformly distributed across the range of the function. This means that Adv's probability of distinguishing the translation of N^0 from N^1 is negligibly small. □

One additional criteria for request privacy is that the response must also be encrypted. If the responses are not weakly private then they may reveal information about the data and application name. Consequently, a request could identify specific application data.

We now make two final remarks with respect to the translation function. First, it must be easily invertible so as to let the producer determine N from $\bar{N}$. Thus, it should be a cryptographic pseudorandom permutation (PRP), which is an invertible PRF. Second, PRFs and PRPs are length-preserving by definition. This means that they are *not uniform*, which violates our previous criteria. There are two ways to make the translation uniform: (1) compute the PRP of the cryptographic hash of N (i.e., $PRF(F(N))$) or (2) pad the input prior to translation. We discuss (2) in Section 7.3. For option (1), it is reasonable to expect the producer to precompute a hash table that maps $F(N)$ to N. This would allow the producer to determine the pre-image of $F(\cdot)$.

4.2.1 Strong Request Privacy

In practice, a PRF is insufficient for strong request privacy. Since PRFs are deterministic, an adversary with oracle access can check the challenger's response and always win. Ideally, the oracle's output should always be randomized and reveal nothing about the input. In practice, this means that the oracle must provide semantic security for its outputs [13]. Such an oracle will never, with overwhelming probability, produce the same output given the same input. This is captured in the following theorem.

THEOREM 5. *Semantically secure encryption is necessary for strong request privacy.*

PROOF. Let $q(\cdot,\cdot)$ be a transformation that is semantically secure [13]. Therefore, access to $\mathcal{O}_{q,i^*}(\cdot)$ does not aid Adv in NameGame. Moreover, since $q(\cdot,\cdot)$ behaves as a PRF, Adv's advantage in distinguishing $\bar{N}^0$ and $\bar{N}^1$ is bounded by $\epsilon(\lambda)$. Therefore, $q(\cdot,\cdot)$ provides strong request privacy. □

5. ON-PATH HBC ADVERSARIES

We now turn to on-path HbC adversaries that can observe requests and responses in transit between consumers and producers.

Recall that communication privacy considers both the request and response of a data exchange. The composition of these elements determines the achievable degree of privacy. In Section 3.3, we showed that weak response privacy requires an IND-secure encryption scheme. Similarly, in Section 3.4, we proved that weak request privacy requires at least a name translation function using a keyed PRF. For strong privacy, requests and responses require a CCA-secure (semantically secure) encryption scheme. These composition rules are reflected in the following corollaries.

COROLLARY 1. *A request-response pair have weak communication privacy if the latter is protected by an IND-secure encryption scheme* **and** *the former is transformed by a keyed PRF.*

COROLLARY 2. *A request-response pair has strong communication privacy if the former is transformed using a CCA-secure encryption scheme* **and** *the latter is protected using a similar scheme.*

Note that our definitions do not depend on the exact capabilities of Adv. For example, consider the strongest adversary that controls *every* router on the path between multiple consumers and a single producer. If requests and responses are strongly private, Adv cannot succeed in a correlation, identification, or leakage attack. Adv can only link the communicating parties. As discussed in Section 3.1, this is more of an issue of anonymity than privacy.

6. AUXILIARY INFORMATION

Weak request and response privacy is attainable only up to a certain extent. If Adv has additional information about requests or responses, it can gain an advantage in weak privacy games. This is because such games permit correlation, which leaks information about content. Suppose that Adv has some *a priori* information about the popularity of some content. Adv can use *frequency* of requested content and *popularity* of known content to gain a non-negligible advantage in winning the game.

In more detail, consider request privacy in the presence of auxiliary information. We assume that Adv can eavesdrop on links and thus learn requests and responses. Let $\mathbb{L}$ be the set of links that Adv controls and let $\mathsf{Peek}(\mathbb{L})$ be a function that returns a packet from one of those links.[5] Let $\mathbb{Q}(N)$ and $\mathbb{R}(N)$ be sets of requests and responses, respectively, for content with name prefix N. We model mappings from $\mathbb{Q}(N)$ to $\mathbb{R}(N)$ as a weighted and directed bipartite graph with edges from $\mathbb{Q}(N)$ to $\mathbb{R}(N)$. An edge (Q, R) with weight l, where $Q \in \mathbb{Q}(N)$ and $R \in \mathbb{R}(N)$, means that request Q returns response R with probability $l/|\mathbb{R}(N)|$. In effect, weight indicates popularity of a given request-response pair.

In this model, request privacy is only possible if the in-degree (deg^-) for every R and the out-degree (deg^+) for every Q are equal **and** the weights of every edge are equal. This means that auxiliary information (frequency and popularity) does not help Adv distinguish between two different responses R and R' since each is equally likely to be the output from a randomly chosen request. This leads to the following observation.

THEOREM 6. *Weak request privacy requires that:*
1. $deg^+(Q_i) = deg^+(Q_j)$ *for all* $Q_i, Q_j \in \mathbb{Q}(N)$.
2. $deg^-(R_i) = deg^-(R_j)$ *for all* $R_i, R_j \in \mathbb{R}(N)$.
3. *The distribution of edges from* $\mathbb{Q}(N)$ *to* $\mathbb{R}(N)$ *is uniform.*
4. *Every edge from* $\mathbb{Q}(N)$ *to* $\mathbb{R}(N)$ *has equal weight.*[6]

[5]Note that this *is not* the same as Adv accessing either oracle in request or response privacy games. Also, Adv can only peek on unencrypted links.

[6]If the weight were not uniform then some responses would be more popular than others. This is what we exploit when constructing a distinguisher in the proof.

PROOF. Assume Adv has additional information about $\mathbb{R}(N)$, e.g., that the edge distribution from $\mathbb{Q}(N)$ to $\mathbb{R}(N)$ is *not uniform*. We need to show that Adv can build a distinguisher $\mathcal{D}$ that wins NameGame with probability greater than $\epsilon(\lambda)$. Let $k = max\{1, \lceil|\mathbb{Q}(N)|/|\mathbb{R}(N)|\rceil\}$, be the expected in-degree for each response $R \in \mathbb{R}(N)$. That is, for a uniform edge distribution, k is the number of requests that map to each response. Let $\mathcal{D}$ be a distinguisher created using $\mathsf{Peek}(\mathbb{L})$ and given to Adv in NameGame. $\mathcal{D}$ observes the network using $\mathsf{Peek}(\mathbb{L})$ to determine the frequency of individual *requests*. Adv runs $\mathcal{D}$ for a polynomial amount of time in order to collect this frequency information. Then, Adv samples two requests (names) Q_0 and Q_1 with the *lowest* and *highest* probabilities, respectively, from its a priori known distribution. Specifically, if $\Pr[Q_i]$ is the popularity of a given name (query) Q_i, then Q_0 and Q_1 are chosen as:

$$Q_0 = \arg\min_{Q_i} \Pr[Q_i]$$
$$Q_1 = \arg\max_{Q_i} \Pr[Q_i]$$

Adv then provides N_0 and N_1 – the names of Q_0 and Q_1 – to the challenger, which responds with N_b. Upon receipt, Adv queries $\mathcal{D}$ with Q_b, obtained from N_b, to determine the relative frequency f of Q_b based on the collected frequency statistics. If $f > k/\mathbb{R}(N)$, then Adv outputs $b' = 1$; otherwise, it outputs $b' = 0$.

The winning probability can be expressed as:

$$\Pr(b' = b) = \Pr(b = 1 \wedge b' = 1) + \Pr(b = 0 \wedge b' = 0)$$

Using Bayes' Theorem, this can be rewritten as:

$$\Pr(b = 1 \wedge b' = 1) + \Pr(b = 0 \wedge b' = 0) =$$
$$\Pr(b = 1)\Pr(b' = 1|b = 1) + \Pr(b = 0)\Pr(b' = 0|b = 0)$$

From the construction of $\mathcal{D}$, it follows that $\Pr(b' = 1|b = 1) = \Pr[Q_1]$ and $\Pr(b' = 0|b = 0) = \Pr[Q_0]$. Let p_1 and p_0 denote these respective probabilities. Also, since b is sampled at random, $\Pr(b = 1) = \Pr(b = 0) = 0.5$.

In order for Adv to win INDNameGame, it must be true that $0.5p_0 + 0.5p_1 > k/\mathbb{D}(N) + \epsilon(\lambda)$. We can rewrite this as $p_0 + p_1 > 2k/\mathbb{D}(N) + \epsilon(\lambda)$, where p_0 and p_1 are the minimum and maximum probabilities in the popularity distribution. Clearly, there are distributions where this inequality holds for any k. One example is the Poisson distribution. □

Under this constraint, weak request privacy is difficult to achieve, since some requests are always more popular than others. Moreover, since content popularity is not controllable by the producer, it cannot expect to enforce uniform popularity.

Implications.

The frequency analysis attack succeeds because responses are not generated with equal probabilities. An intuitive way to mitigate the attack is to ensure that every response is unique. One way to realize this is by requiring a producer to encrypt each response separately and uniquely. However, this immediately obviates the main benefit of caching.

An alternative is for a router to individually re-encrypt cached content before passing it downstream. (Thus, it would be infeasible to correlate content coming into a router with another content later leaving the same router.) Unfortunately, this seems to be possible only via so-called *proxy re-encryption* or re-randomizable encryption techniques which are typically based on public key schemes, e.g., ElGamal. More importantly, these techniques would require *the entire content* to be re-randomized, which is likely to be prohibitively expensive. Specifically, this would rule out the use of traditional (and efficient) hybrid encryption. (In hybrid encryption, data is encrypted using an efficient symmetric cipher with a one-time key, and only the latter is encrypted using expensive public key encryption under the public key of the decryptor). Therefore, bulk content encryption, re-encryption and eventual decryption would all have to be performed using purely public key cryptography.

Even if the above were not an issue, there would remain a problem due to requests for various names not being uniformly distributed. One counter-measure is to encrypt each request using a CPA-secure encryption scheme. Then, two requests for the same content would carry different names that cannot be correlated. This would, once again, negate the main benefit of caching. Thus, it appears that weak privacy – in the presence of Adv that has auxiliary information – is only attainable if both requests and responses are individually encrypted, making router caching useless, except for re-transmissions.

7. PRIVACY IN PRACTICE

We now discuss application design patterns that can achieve levels of privacy discussed above. However, since we can not view requests and responses in isolation, we only focus on design patterns for communication privacy. This is because most realistic eavesdropping adversary would be capable of observing bi-directional traffic, i.e., both requests and responses.

7.1 Assumptions

As discussed earlier, a PRF or semantically secure encryption scheme is needed for request privacy. Hence, the consumer must have some information about the producer before issuing the request. Clearly, a request transformation function is only useful if the producer can *efficiently* compute its inverse. Also, we assume that the name transformation index, i.e., the minimal routable prefix, is known to both consumer and producer.

7.2 Weak Privacy

Recall that weak privacy allows correlation and no identification or leakage. This degree of privacy is unsuitable for highly sensitive data exchanges, such as online banking or e-commerce transactions. However, it might be suitable for less sensitive applications, e.g., content distribution networks (CDNs) which distribute *static* content. Concrete examples include Netflix, Spotify, Imgur and Flickr. Applications can use CDNs by asking for content with the same name. As long as a name does not reveal any information about the corresponding content, the CDN does not need to know to what data it refers. A CDN node maps a request to a response based on exact name match. We believe that this general model is a good choice for weak communication privacy, wherein requests are protected by a keyed PRF and responses – by an IND-secure encryption scheme.

To support this type of privacy, there are two cases in terms of the producer and consumer(s) relationship. In the first one, they share a secret such as a previously established key. Let k be a unique key derived (e.g., via a PRF) from this shared secret. We could then instantiate $q(\cdot,\cdot)$ and $r(\cdot,\cdot)$ as an IND-secure encryption scheme based on a PRP indexed by k.

Now suppose the producer and consumer have no pre-shared secrets. Then, before requesting content from a producer, a consumer must know the former's public key. In this case, $q(\cdot,\cdot)$ can be any IND-secure (and thus CPA-secure) public key encryption function, e.g., RSA. The response must be likewise protected using IND-secure encryption. One way is to encrypt it using the consumer's public key, which the consumer would include in the request. Alternatively, the consumer could pick a one-time symmetric key and also include it in the original request. Of course, this does not provide forward security, which is another aspect to consider.

This is clearly an inefficient solution since it effectively removes the utility of router caches, which is the primary reason one would choose weak privacy. A better approach would be to encrypt the content under a broadcast encryption scheme, e.g., [4]. Such schemes are CCA-secure and therefore suitable for our setting since IND-secure encryption follows from CCA-security. However, they only work if the consumer already has the decryption keys, which violates our assumption that consumers and producers have no pre-shared secrets. Therefore, it is unclear how to enable efficient weak privacy with caching for *public content*, i.e., content that can be requested by anyone without authentication or authorization.

7.3 Name Padding

Since IND-secure encryption is required for weak privacy, we must also address the issue of name padding to make these transformations uniform. Theorem 3 states that every encrypted name in a given namespace must be of the same length. This is only possible if all names are padded to some maximal length. The current CCN packet format limits the total name length to 64KB [18]. In practice, names are much smaller. To verify this, we analyzed names in the Unibas dataset from The Content Name Collection [1], which contains *unique* URLs submitted by users to URL shortner websites. We converted these URLs into a CCN-compatible name format. For example, the URL `http://www.domain.com/file.html` is converted into `/com/domain/file.html`. Table 2 shows some characteristics of the Unibas dataset. The standard deviation of the number of segments in a name is 8.14 and the mean length per segment is 10.39B. Even with these smaller sizes, performing padding universally across all namespaces would result in significant overhead: the size of an encrypted name would be about 800KB (based on the maximum segment length, which is well beyond the maximum threshold for CCN packets). For links with small(er) MTUs, e.g., in the range $[1,500\text{B},\, 9,000\text{B}]$, the performance impact would be very heavy since it would almost always induce fragmentation. Even with secure fragmentation schemes such as those in [9] and [20], the overhead would be non-negligible.

Fortunately, names are not distributed uniformly. Table 3 illustrates the name distribution per number of segments in each name. It shows the number of names in the dataset that contain n segments for $n \in [1, 20]$. Note that: (1) almost 30% of names have 5 segments, and (2) names of up to 20 segments account for 99.876% of the dataset. Thus, maximum padding size is much smaller than the maximum name size.

Furthermore, padding could be applied on a per-namespace basis. For example, in namespace `/`, the maximum padding length is around 800KB. However, under the namespace `/netflix/`, the maximum name length is likely much smaller. Since an application has complete control over its namespace, it can specify a maximum length for consumers to use in padding.

As a final note, this requires i^* to be as long as the minimal routable (un-encrypred) prefix needed to forward encrypted requests to the producer. For example, suppose that all interests in the `/netflix/` namespace were routable. We would need to set $i^* = 1$, since anything beyond that would leak information about the request. This highlights the relationship between namespace ownership, routing, and privacy.

7.4 Strong Privacy

Strong privacy is suitable for applications for which security is more important than caching. Examples include: banking, e-

Table 2: Unibas Dataset Characteristics

Names		**Name segments**		**Segments per Name**	
Number of names	$870'896'633$	Total number of segments	$4'855'203'042$	Total number of segments	$4'855'203'042$
Average name length (bytes)	57.95	Average segment length (bytes)	10.39	Average segments per name	5.57
Name length standard deviation	77.60	Segment length standard deviation	30.02	Segments per name standard deviation	8.14
Minimum name length (bytes)	1	Minimum segments length (bytes)	1	Minimum segments per name	1
Maximum name length (bytes)	$\mathbf{764'867}$	Maximum segments length (bytes)	$\mathbf{764'867}$	Maximum segments per name	$210'658$

Table 3: Name Distribution per # of Segments

Number of segments n	**Number of names**	**Percentage**
1	$13'952$	0.002%
2	$141'904$	0.016%
3	$71'327'647$	8.190%
4	$187'307'048$	21.507%
5	$253'852'565$	29.148%
6	$144'130'578$	16.550%
7	$93'837'904$	10.775%
8	$70'875'144$	8.138%
9	$25'611'959$	2.941%
10	$10'464'092$	1.202%
11	$3'973'961$	0.456%
12	$4'546'842$	0.522%
13	$1'206'905$	0.139%
14	$835'124$	0.096%
15	$844'552$	0.097%
16	$195'491$	0.022%
17	$121'486$	0.014%
18	$317'628$	0.036%
19	$168'228$	0.019%
20	$50'742$	0.006%
Total	$\mathbf{869'823'752}$	$\mathbf{99.876\%}$

commerce, and voting. Fortunately, in such cases, there are fewer design decisions, since both requests and responses require semantic or CCA-secure encryption. This immediately implies the need for a session-based protocol in which *no* request and response can be correlated. There is only one such protocol for ICNs – CCNx Key Exchange (CCNxKE) [25]. It is inspired by TLS 1.3 [22] – the latest version of the Transport Layer Security protocol. Assuming only knowledge of the minimal routable prefix for a given of content, CCNxKE allows consumers and producers to create a secure session with forward-secure keys to encrypt both requests and responses using a CCA-secure encryption scheme. Clearly, strong privacy destroys any benefits of shared caching for consumers. We believe this is an important takeaway from our work.

8. RELATED WORK

Privacy in CCN and related architectures was initially addressed in [5]. It briefly discusses the concepts of content and name privacy. However, it does not consider these problems in any formal adversarial model. Moreover, some of the solutions for name privacy induce architectural changes. In this work, we assessed privacy in the context of the standard CCN architecture.

Response privacy has received considerable attention in the research community. Generally, responses are encrypted such that only authorized consumers may decrypt the contents. This type of technique permits content to be disseminated throughout the network since it cannot be decrypted without the appropriate decryption key(s). Many variations of this approach have been proposed based on general group-based encryption [23], broadcast encryption [15], attribute-based encryption [11], and proxy re-encryption [26]. Kurihara et al. [14] generalized these specialized approaches in a framework called CCN-AC, an encryption-based access control framework to implement, specify, and enforce access policies. It uses CCN manifests[7] to encode access control specification information for a particular set of content objects. Consumers use information in the manifest to (1) request appropriate decryption keys and (2) decrypt the content object(s) in question. The NDN NBAC [27] scheme is similar to [14] in that it allows decryption keys to be flexibly specified by a data owner. However, this is done based on name convention rules instead of configuration. [27] showed that this leads to better overall performance and scalability when compared to CCN-AC. Unfortunately, all the aforementioned techniques require at least one interest to fetch the content decryption key. Even if the primary content object request was weakly private, the request for the decryption key is not since it must identify the key for the requesting consumer.

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. The proposed technique is only weakly private since they were designed with cache utility in mind. Specifically, two consumers in the same "access control group" issue identical requests for the same protected content.

Privacy issues outside of the core request and response protocol have been explored in the context of caches [2, 16, 17]. 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.

9. CONCLUSION AND FUTURE WORK

We presented a comprehensive assessment of data privacy in CCN based on the constituent privacy properties of requests and responses. We described the properties of requests and responses necessary to achieve certain types of privacy in the presence of various adversaries. We show that strong cryptographic protection is necessary for both requests and responses. As future work, we plan

[7]Manifests are special types of content that are used to provide structure and additional information to otherwise flat and simple content objects [19].

to examine application communication patterns to learn what is leaked from side-channels. For example, protecting against timing and size side-channels might require even stronger protection mechanisms beyond what is proposed in this paper.

10. REFERENCES

[1] The content name collection. http://www.icn-names.net/. Accessed: August 3, 2016.

[2] G. Acs, M. Conti, P. Gasti, et al. Cache privacy in named-data networking. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pages 41–51. IEEE, 2013.

[3] T. Berners-Lee, R. Fielding, and L. Masinter. RFC 2396: Uniform resource identifiers (URI): generic syntax, 1998.

[4] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology–CRYPTO 2005*, pages 258–275. Springer, 2005.

[5] A. Chaabane, E. De Cristofaro, M. A. Kaafar, et al. Privacy in content-oriented networking: Threats and countermeasures. *ACM SIGCOMM Computer Communication Review*, 43(3):25–33, 2013.

[6] A. Cooper, H. Tschofenig, B. Aboba, et al. Privacy considerations for internet protocols. *Internet Architecture Board*, 2013.

[7] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. Andana: Anonymous named data networking application. In *NDSS*, 2011.

[8] S. Farrell and H. Tschofenig. Pervasive monitoring is an attack. 2014.

[9] C. Ghali, A. Narayanan, D. Oran, et al. Secure fragmentation for content-centric networks. In *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*, pages 47–56. IEEE, 2015.

[10] C. Ghali, M. A. Schlosberg, G. Tsudik, et al. Interest-based access control for content centric networks. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 147–156. ACM, 2015.

[11] M. Ion, J. Zhang, and E. M. Schooler. Toward content-centric privacy in icn: Attribute-based encryption and routing. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 39–40. ACM, 2013.

[12] V. Jacobson, D. K. Smetters, J. D. Thornton, et al. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[13] J. Katz and Y. Lindell. *Introduction to modern cryptography: principles and protocols*. CRC press, 2007.

[14] J. Kurihara, C. Wood, and E. Uzuin. An encryption-based access control framework for content-centric networking. *IFIP*, 2015.

[15] S. Misra, R. Tourani, and N. E. Majd. Secure content delivery in information-centric networks: Design, implementation, and analyses. In *ICN*, 2013.

[16] A. Mohaisen, H. Mekky, X. Zhang, et al. Timing attacks on access privacy in information centric networks and countermeasures. 2015.

[17] A. Mohaisen, X. Zhang, M. Schuchard, H. Xie, and Y. Kim. Protecting access privacy of cached contents in information centric networks. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 173–178. ACM, 2013.

[18] M. Mosko, I. Solis, and C. Wood. CCNx Messages in TLV Format. Internet-Draft draft-irtf-icnrg-ccnxmessages-03, Internet Engineering Task Force, June 2016. Work in Progress.

[19] M. Mosko, I. Solis, and C. Wood. CCNx Semantics. Internet-Draft draft-irtf-icnrg-ccnxsemantics-03, Internet Engineering Task Force, June 2016. Work in Progress.

[20] M. Mosko and C. A. Wood. Secure fragmentation for content centric networking. In *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on*, pages 506–512. IEEE, 2015.

[21] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):12–19, 2002.

[22] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. Internet-Draft draft-ietf-tls-tls13-14, Internet Engineering Task Force, July 2016. Work in Progress.

[23] D. K. Smetters, P. Golle, and J. D. Thornton. CCNx access control specifications. Technical report, PARC, July 2010.

[24] G. Tsudik, E. Uzun, and C. A. Wood. AC3N: Anonymous communication in Content-Centric Networking. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 988–991, Jan 2016.

[25] C. Wood, M. Mosko, and E. Uzun. CCNx Key Exchange Protocol Version 1.0. Internet-Draft draft-wood-icnrg-ccnxkeyexchange-00, Internet Engineering Task Force, July 2016. Work in Progress.

[26] C. A. Wood and E. Uzun. Flexible end-to-end content security in CCN. In *CCNC*, 2014.

[27] Y. Yu, A. Afanasyev, and L. Zhang. Name-based access control. *Named Data Networking Project, Technical Report NDN-0034*, 2015.

[28] L. Zhang, A. Afanasyev, J. Burke, et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.