Interest-Based Access Control for Content-Centric Networks

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

ABSTRACT

Content-Centric Networking (CCN) is an emerging network architecture designed to overcome limitations of the current IP-based Internet. One of the fundamental tenets of CCN is that content is named and addressable. Consumers request content by issuing interests with the desired content name. These interests are forwarded by routers to producers, and the requested content is returned and optionally cached at each router along the path.

In-network caching makes it difficult to enforce access control policies on sensitive content since routers only use interest information for forwarding decisions. This motives our work on Interest-Based Access Control (IBAC) – a scheme for access control enforcement using only information contained in interest messages. IBAC makes sensitive content names unpredictable to unauthorized parties. It supports both hash- and encryption-based name obfuscation. Interest replay attacks are addressed by formulating a mutual trust framework between producers and consumers that enables routers to perform authorization checks before satisfying interests from local caches. We assess computational, storage, and bandwidth costs of each IBAC variant. Proposed design is flexible and allows producers to arbitrarily specify and enforce any type of content access control, without having to deal with content encryption and key distribution. This is the first comprehensive CCN access control design that only uses information contained in interest messages.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network communications*

[*]Supported by the NSF grant: "CNS-1040802: FIA: Collaborative Research: Named Data Networking (NDN)".

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

ICN'15, September 30–October 2, 2015, San Francisco, CA, USA.
© 2015 ACM. ISBN 978-1-4503-3855-4/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810156.2810174 .

General Terms

Theory, Design, Performance

Keywords

content-centric networks; access control; name obfuscation

1. INTRODUCTION

The purpose of the original Internet in the 1970-s was to provide end-to-end communication for a few thousand users to access scarce and expensive resources via terminals. Since then, the number of Internet users has grown exponentially, exceeding 3 billion, each using a wide variety of applications: from dynamic web to content distribution. This shift of usage exposed certain limitations of the IP-based Internet design and motivated exploration of new Internet architectures.

Content-Centric Networking (CCN) is an approach to inter-networking exemplified by two well-known research efforts: CCNx [19] and Named-Data Networking (NDN) [12]. CCN's main goal is to develop the next-generation Internet architecture with an emphasis on efficient content distribution, security, and privacy. Unlike current IP-based networking where data is requested by addressing the machine hosting the data, each CCN *content* is assigned a unique name. Users (called consumers) request content objects by issuing an *interest* for a given name. This interest can then be satisfied or served from any entity (i.e., producer or router) as long as the replied content's name matches that of the interest.

To facilitate efficient content distribution, a CCN router maintains a cache. This enables routers to satisfy interests, which reduces end-to-end latency and decreases bandwidth utilization when requesting popular content. Since interest messages may be satisfied by any cached version of the content, interest messages may not, and need not, reach the producer. Therefore, enforcing content access control *within the network* is a challenge. Furthermore, even if all interests are forwarded to producers, the latter might not be able to enforce access control since interest messages, by design, do not carry any form of consumer identification or authentication information.

In this paper, we propose an access control scheme based on interests – Interest-Based Access Control (IBAC). The intuition is that if consumers are not allowed access to certain content, they should not be able to generate the corresponding interests, i.e., they should not be able to learn the content's name. IBAC may also be used with or alongside content encryption to conceal both the name and the payload of the content object.[1] However, using IBAC in isolation is advantageous in scenarios where content object payloads may need to be modified by an intermediary service, e.g., a media encoding application or proxy. In this case, content encryption prevents such modifications by services or applications besides the producer. Moreover, although IBAC involves the network layer, we believe that this is necessary to allow routers (with caches) to enforce access control. To be more specific, we claim that, to support IBAC, *any entity which serves content should also be able to authorize interests for said content.*

The main contributions of this paper are:

- Architectural modifications to support IBAC without diminishing caching benefits.
- A mutual trust scheme wherein routers can verify whether consumers are authorized to access cached content.
- A security analysis of the proposed IBAC scheme.
- Evaluation of router performance overhead when serving content via IBAC compared to publicly accessible content.

The rest of this paper is organized as follows. Section 2 presents a CCN overview. Then, Section 3 provides an summarizes CCN access control techniques. Next, Section 4 presents security definitions and the adversary model, followed by Section 5 that presents the IBAC scheme. Security considerations are discussed in Section 6 and IBAC costs are assessed in Section 7. The paper concludes in Section 8.

2. CCN OVERVIEW

Content Centric Networking (CCN) is one of the main Information-Centric Networking (ICN) architectures. Related architectures, such as Named Data Networking (NDN) [25], are similar, albeit with some small protocol and packet format differences. This section overviews ICNs in the context of the CCN protocol and CCNx reference implementation. Given familiarity with either CCN or NDN, it can be skipped without loss of continuity.

In contrast to TCP/IP, which focuses on end-points of communication and their names and addresses, ICN architectures such as CCN [12, 19] focus on content by making it named, addressable, and routable within the network. A content name is a URI-like [2] name composed of one or more variable-length name components, each separated by a `/` character. To obtain content, a user (consumer) issues a request, called an *interest* message, with the name of the desired content. This interest will be *satisfied* by either (1) a router cache or (2) the content producer. A *content object* message is returned to the consumer upon satisfaction of the interest. Moreover, name matching in CCN is exact, e.g., an interest for `lci:/facebook/Alice/profile.html` can only be satisfied by a content object named `lci:/facebook/Alice/profile.html`.[2]

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

- `Payload` – enables consumers to push data to producers along with the request.[3]
- `KeyID` – an optional hash digest of the public key used to verify the desired content's digital signature. If this field exists, the network guarantees that only content objects which can be verified with the specified key will be returned in response to an interest.
- `ContentObjectHash` – 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.

CCN content objects include several fields. In this work, we are only interested in the following three:

- `Name` – a URI-like name formatted as a sequence of `/`-separated name components.
- `Validation` – a composite of validation algorithm information (e.g., the signature algorithm used, its parameters, and a link to the public verification key), and validation payload (e.g., the signature). We use the term "signature" to refer to this field.
- `ExpiryTime` – an optional, producer-recommended time for the content objects to be cached.

There are three types of entities in CCN:[4]

- *Consumer* – issues an interest for content.
- *Producer* – produces and publishes content.
- *Router* – routes interest packets and forwards corresponding content packets.

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-matches 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 content 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.

Routers use the FIB to forward interests from consumers to producers, and the PIT – to forward content object messages along the reverse path to the consumer. More specifically, upon receiving an interest, a router R first checks its cache to see if it can satisfy this interest locally. Producer-originated digital signatures allow consumers to authenticate received content, regardless of the entity that actually served the content. Moreover, the Interest-Key Binding rule

[1]As we will discuss, IBAC does not replace access control based on content encryption. It is a complementary form of access control.

[2]Name matching is not exact in NDN [25].

[3]Currently, NDN interest messages do not provide an arbitrary-length payload and therefore cannot support the proposed IBAC scheme. However, if in the future the NDN interest format is modified to include a field similar to the CCNx payload, our IBAC scheme will become applicable.

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

(IKB) [9] enables routers to efficiently verify received content signatures before caching, in order to avoid content poisoning attacks [8]. Essentially, consumers and producers provide routers with the required trust context to enable efficient signature verification.

When a router R receives an interest for name N that is not cached 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 amount of state information, 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 deemed unsolicited and subsequently discarded.

3. ACCESS CONTROL OVERVIEW

One key feature of CCN is that content is decoupled from its source; there is no notion of a secure channel between a consumer and producer. Consequently, ensuring that only authorized entities have access to content is a fundamental problem. In this section, we explore complementary approaches to access control: (1) content encryption and (2) interest name obfuscation and authorization.

3.1 Encryption-Based Access Control

The most intuitive solution to the access control problem is via encrypted content which can only be decrypted by authorized consumers possessing the appropriate decryption key(s). This enables content objects to be disseminated throughout the network since they cannot be decrypted by adversaries without the appropriate decryption key(s).

Many variations of this approach have been proposed [21, 18, 11, 23]. Kurihara et al. [17] 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[5] 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) use them to decrypt the content object(s) in question.

Outside of ICN, there have been many proposed access control frameworks based on encryption. Recently, access control in shared cloud storage or social network services, e.g., Google Drive, Dropbox, and Facebook, have generated much attention from the research community [26, 22, 24, 13]. For instance, Kamara et al. [14] modeled encryption-based access control framework for cloud storage. Microsoft PlayReady [1] is another popular access control framework for encrypted content dissemination over the Internet.

Despite its widespread use, encryption-based access control causes potentially prohibitive overhead for both producers and consumers. In most cases where hybrid encryption is used, it also requires keys to be distributed alongside each content object, which introduces another consumer-to-producer message exchange. Also, encryption-based access control does not provide flexibility if content objects need to be modified by an intermediate service, e.g., a media encoding or enhancement application. Content encryption prevents such post-publication modifications without revealing the secret decryption key(s) to such services.

3.2 Interest-Based Access Control

Interest-based access control (IBAC) is an alternative technique, though not mutually exclusive with content encryption, for implementing access control in CCN. It is based on interest *name obfuscation* and *authorized disclosure*. Name obfuscation hides the *target* of an interest from eavesdroppers. As mentioned in [12], name obfuscation has no impact on the network since routers use only the binary representation of a name when indexing into PIT, CS, and FIB. As long as producers generate content objects with matching names, the network can seamlessly route interests and content objects with obfuscated names. However, interests with obfuscated names *must contain routable prefixes* so that the interests can still be forwarded from consumers to the producers. In other words, only a subset of name components (e.g., the suffix of the name) can be obfuscated.

Another goal of name obfuscation is to *prevent unauthorized users from creating interests for protected content*. In other words, if a particular consumer Cr is not permitted to access content with name N, Cr should not be able to generate $N' = f(N)$, where $f(\cdot)$ is some obfuscation function that maps N to an obfuscated name N'. For routing purposes, only the *suffix* of the name is obfuscated; there must exist a cleartext prefix that is used to route the interest with a partially obfuscated name to the intended producer. Possible obfuscation functions include keyed cryptographic hash functions and encryption algorithms. We explore both possibilities in this paper.

Authorized disclosure is the second element of IBAC. This property implies that any entity serving content must authorize any interest for said content before it is served. In this context, authorization is necessarily coupled with authentication so that the entity serving the content can determine the identity of the requesting consumer. Therefore, consumers must provide sufficient authentication information, e.g., via an interest signature. Thus, to implement authorized disclosure (in the presence of router caches), any entity serving content must (a) possess the information necessary to perform authentication and authorization checks and (b) actually verify the provided authentication information. This issue is discussed at length in Section 6.2. It is worth mentioning that disabling content caching defers authorized disclosure checks to producers. In this case, all interests will be forwarded to producers that posses the information needed to perform these checks. However, by itself, prohibiting content from being cached is *not* a form of access control and reduces the effectiveness of content retrieval.

Fotiou et. al. [5] proposed an access control mechanism similar to IBAC for non-ICN architectures, and conjectured that it should be applicable to ICNs. In [5], access control computation and overhead are delegated to a separate, non-cache entity. This entity, known as the access control provider, maintains access control policies given by a specific producer. Each content object has a pointer to a function that determines whether or not to serve the content to the requesting consumer, and the access control provider

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

is responsible for evaluating this function. Content objects are stored at relaying parties, which are oblivious to the specific access control policy protecting the content objects. Similarly, the access control provider has no knowledge of the consumer requesting the content (for user privacy purposes), and just evaluates whether the relaying party should forward the content object. The cache, in this scenario, is not responsible for the extra computational overhead. This approach is different from our work in that we (1) maintain the association between content and authorization, and (2) provide routers with an efficient authorization verification method, thus eliminating the need for an external access control provider.

4. SECURITY MODEL

Let $\mathbb{U}(N)$ denote the set of authorized consumers for a content object with name N generated and controlled by a producer P, and let $\bar{\mathbb{U}}(N)$ be its complement, i.e., the set of all unauthorized consumers. Let $\mathsf{Path}(Cr, P)$ be the set of all routers on the path between the consumer $Cr \in \mathbb{U}(N)$ and P. We assume the existence of an adversary Adv who can deploy and compromise any router $R \notin \mathsf{Path}(Cr, P)$.[6] To keep this model realistic, we assume that the time to mount such an attack is non-negligible, i.e., longer than the average RTT for a single interest-content exchange. Table 1 summarizes the notation used in the rest of this paper.

Formally, we define Adv as a 3-tuple: $(\mathcal{P}_{\text{Adv}} \setminus \{P\}, \mathcal{C}_{\text{Adv}} \setminus \mathbb{U}(N), \mathcal{R}_{\text{Adv}} \setminus \mathsf{Path}(Cr, P))$ where the components denote the set of compromised producers, consumers, and routers, respectively. If Adv controls a producer or a consumer then it is assumed to have complete and adaptive control over how they behave in an application session. Moreover, Adv can control all of the timing, format, and actual information of each content through compromised nodes and links.

Let Guess denote the event where Adv correctly recovers the obfuscated form of a content name. Let Bypass denote the event where Adv successfully bypasses the authorization check for a protected content object. We define the security of an IBAC scheme with respect to these two events as follows.

Definition 1. An IBAC scheme is *secure, but subject to replay attacks,* if $\Pr[\mathsf{Guess}] \leq \epsilon(\kappa)$ for any negligible function ϵ and a security parameter κ.

Definition 2. An IBAC scheme is *secure in the presence of replay attacks,* if $\Pr[\mathsf{Guess} + \mathsf{Bypass}] \leq \epsilon(\kappa)$ for any negligible function ϵ and a security parameter κ.

Replay attacks are artifacts of the environment when a CCN access control scheme is deployed. In other words, in networks where links are insecure, passive eavesdroppers can observe previously issued interests and replay them for protected content. Consequently, these attacks are considered orthogonal to the security of the underlying obfuscation scheme used for access control enforcement. The authorized disclosure element of IBAC is intended to prevent such replay attacks.

To justify our adversarial limitation to off-path routers, consider the following scenario. If Adv can compromise a router $R \in \mathsf{Path}(Cr, P)$, then Adv is able to observe *all* content that flows along this path. Therefore, we claim that on-path adversaries motivate access control schemes based on content encryption; as such, IBAC will not suffice. Moreover, we exclude adversaries capable of capturing interests and replaying them in other parts of the network – see Section 6.1 for details.

[6]Any one of these actions can be performed adaptively, i.e., in response to status updates or based on observations.

Table 1: Relevant notation.

Notation	Description
Adv	Adversary
Cr	Consumer
P	Producer
prefix	Producer prefix
N	Content name in cleartext
N'	Obfuscated content name
$I[N]$	Interest with name N
CO	Content object
$CO[N]$	Content object with name N
$\mathsf{ID}(\cdot, \cdot)$	Key identifier function
$f(\cdot)$	Obfuscation function
$\mathsf{enc}(\cdot, \cdot), \mathsf{dec}(\cdot, \cdot)$	Symmetric-key encryption and decryption function
$\mathsf{Enc}(\cdot, \cdot), \mathsf{Dec}(\cdot, \cdot)$	Public-key encryption and decryption function
$\mathsf{H}(\cdot)$	Cryptographic hash function
$\mathbb{U}(N)$	Set of authorized consumers
$\mathbb{G}_i$	Access control group i
$k_{\mathbb{G}_i}$	Obfuscation key of group $\mathbb{G}_i$
$pk^s_{\mathbb{G}_i}, sk^s_{\mathbb{G}_i}$	Public and private signing key pair associated with group $\mathbb{G}_i$
κ	Global security parameter
$\mathbb{C}$	Set of all content objects
r, t	nonce and timestamp
$\mathbb{B}$	Nonce hash table

5. IBAC BY NAME OBFUSCATION

Recall that the intuition behind IBAC is that if consumers are not allowed to access certain content, they should not be able to issue a "correct" interest for it. Specifically, only a consumer $Cr \in \mathbb{U}(N)$ should be able to derive the obfuscated name N' of an interest requesting content with name N provided by producer P. In this section, we discuss two types of name obfuscation functions: (1) encryption functions and (2) hash functions.

5.1 Encryption-Based Name Obfuscation

Let $\mathsf{Enc}(k, N)$ be a *deterministic* encryption function which takes as input a key $k \in \{0, 1\}^\kappa$ and an arbitrary long non-empty binary name string N, and generates an encrypted name N'. Let $\mathsf{Dec}(k, N')$ be the respective decryption function. With encryption, the goal is for authorized clients to encrypt components of a name so that the producer can perform decryption to identify and return the appropriate content object.[7] Obfuscation is based on knowledge of the encryption key and the content name under IBAC protection. In other words, even if an adversary knows the name N, it cannot generate N' since it does not possess the appropriate key.

[7]Recall that a cleartext name prefix is needed to route the interest to the intended producer.

To illustrate how encryption-based obfuscation would work, assume first that Cr uses k to generate N' as $N' = \mathsf{Enc}(k, N)$. P then recovers N as $N = \mathsf{Dec}(k, N')$ to identify the content object in question and returns it with the *matching name* N' (not N). We prove the security of this obfuscation variant of IBAC (i.e., without authorized disclosure) in [7].

Supporting Multiple Access Groups: Thus far, we assumed that name encryption (obfuscation) keys are known to all authorized consumers in $\mathbb{U}(N)$. However, this might not be the case in practice. P might provide content under IBAC to several access groups each with different privileges.[8] Specifically, consumers in groups $\mathbb{G}_i(N) \subset \mathbb{U}(N)$, for $i = 1, 2, \ldots,$ might be allowed access to different resources. Therefore, several obfuscation keys, one for each group, should be utilized. For notation simplicity, we refer to $\mathbb{G}_i(N)$ as $\mathbb{G}_i$. Note that in an extreme scenario, each group would only contain a single consumer, i.e., each individual consumer has a unique key used to access the content in question.

To decrypt the obfuscated name N', P must identify the obfuscation key used to generate N'. This can be achieved if such consumers specify an identifier for the key used in the interest. Such an identifier could simply be the digest of the obfuscation key $\mathsf{ID}_{\mathbb{G}_i} = \mathsf{H}(k_{\mathbb{G}_i})$, where $k_{\mathbb{G}_i}$ is $\mathbb{G}_i$'s encryption key. $\mathsf{ID}_{\mathbb{G}_i}$ can be included in the interest `Payload` field. Since matching in CCN is exact, $\mathsf{ID}_{\mathbb{G}_i}$ cannot be included in the interest name.

Recall that CCN interest messages, by design, do not carry any source information, which provides some degree of anonymity. However, including $\mathsf{ID}_{\mathbb{G}_i}$ enables interest linkability by eavesdroppers (malicious or not). In other words, $\mathsf{ID}_{\mathbb{G}_i}$ can reveal the access group identities to which consumers belong, but not the identities of the consumers themselves. If this linkability is an issue for applications, $\mathsf{H}(k_{\mathbb{G}_i})$ can be encrypted using P's public key pk^P in the form $\mathsf{ID}_{\mathbb{G}_i} = \mathsf{Enc}(pk^P, \mathsf{H}(k_{\mathbb{G}_i}))$.[9] Note that for two identifier values of the same group, i.e., with the same k, to be indistinguishable, $\mathsf{Enc}(\cdot, \cdot)$ must be secure against chosen plaintext attacks [15].

5.2 Hash-Based Name Obfuscation

Let $\mathsf{H}(k, N)$ be a keyed cryptographic hash function. The obfuscated name N' can be generated as $N' = \mathsf{H}(k, N)$ for some key $k \in \{0, 1\}^\kappa$. Since hash functions are one-way, producers must maintain a hash table that maps obfuscated names to the original content name, i.e., $\mathsf{M} : N' = H(k, N) \to N$ for all deployed keys.[10] The size of this hash table is $\mathcal{O}(|\mathbb{K}| \times |\mathbb{C}|)$, where $\mathbb{K}$ is the set of all keys and $\mathbb{C}$ is set of all content objects generated or published by P under IBAC protection. This approach provides the same benefits of encryption-based name obfuscation, however, it incurs additional computation and storage overhead at the producer. Thus, while keyed hash functions are viable for name obfuscation, deterministic encryption is a much better approach.

[8]We assume that each content object is only accessible by a single access group. However, this assumption will be relaxed later in the paper.

[9]Since a consumer cannot be expected to know the router from which content will be served, it is not plausible for them to encrypt these IDs with the public key of a (set of) router(s).

[10]Producers do not have to keep hash tables for all *possible* keys of size κ, only tables of keys used by producers and issued to access groups.

6. SECURITY CONSIDERATIONS

In this section we discuss the security of IBAC with respect to the adversary model described in Section 4.

6.1 Replay Attacks

Regardless of the obfuscation function used, both previously described IBAC schemes are susceptible to replay attacks. This is because both obfuscation functions are deterministic. Therefore, an eavesdropper $\text{Adv} \in \bar{\mathbb{U}}(N)$ could issue an interest with a captured N' and receive the corresponding content under IBAC protection from either the producer or a router cache. In other words, the same "feature" that makes it possible for authorized consumers to fetch IBAC-protected content from router caches also makes it susceptible to replay attacks.

Such replay attacks are problematic in many access control systems. Standard countermeasures include the use of random, per-message nonces or timestamps. Nonces help ensure that each message is unique, whereas timestamps protect against interests being replayed at later points in time. Thus, to mitigate replay attacks, we use both nonces and timestamps. In particular, each consumer $Cr \in \mathbb{U}(N)$ must issue an interest with (1) name N', (2) a randomly generated nonce r, and (3) a fresh timestamp t. The reason why we use both nonces and timestamps is to allow for loosely synchronized clocks and unpredictable network latencies. Note that if (1) clocks of consumers, producers, and involved routers in IBAC can be perfectly synchronized, and (2) network latencies can be accurately predicted, only timestamps are sufficient for replay detection. Moreover, since nonces and timestamps serve a purpose which is orthogonal to content identification and message routing, they are included in the interest payload.

Consumer nonces are random κ-bit values. If a router receives a duplicate nonce, it can safely assume that the corresponding interest is replayed and should be dropped. Let w be a time window associated with authorized content.[11] To determine if a duplicate nonce was received, producers (or caches) must maintain a collection of nonces for each such content. In other words, this historical information is necessary to prevent replay attacks. Timestamps themselves are not stored, they are only used to determine if the received interest is issued within the acceptable time window w. Once this time window elapses, all of the stored nonces are erased and the content is subsequently flushed from the cache.

Although using nonces and timestamps allows detection of replayed interests, Adv capturing interests can still use their obfuscated names N' to fabricate another interest with legitimate r and t values. Therefore, we also stipulate that r and t should be authenticated via a digital signature; their signature σ is also included in the interest `Payload` field. In order to bind r and t to their corresponding interest, N' is also included in the signature computation. σ generation

[11]Determining the proper value of w is outside the scope of this paper. However, a logical approach is for routers to use the lifetime of authorized content as w.

and verification should be performed using the public and private key pairs associated with each access group $\mathbb{G}_i$.

After adding nonces, timestamps, and a signature, interest `Payload` fields take the following form:

$$\texttt{Payload} = \left(\mathsf{ID}_{\mathbb{G}_i}, r, t, \sigma = \mathsf{Sign}_{sk^s_{\mathbb{G}_i}}\left(N'||\mathsf{ID}_{\mathbb{G}_i}||r||t\right)\right)$$

where $\mathsf{ID}_{\mathbb{G}_i}$ is the identity of group $\mathbb{G}_i$, and $sk^s_{\mathbb{G}_i}$ is a signing key distributed to all consumers in $\mathbb{G}_i$. To verify σ, the matching public key $pk^s_{\mathbb{G}_i}$ must be obtained. For the remainder of this paper, we use the term *authorization information* to refer to all information included in interest `Payload` fields for the purpose of supporting IBAC.

One alternative to digital signatures would be to use a keyed hash or a Message Authentication Code function such as (HMAC) [16]. In this case, consumers and routers would need to share the key used in the HMAC computation. This means that either consumers or producers need to distribute HMAC keys to all involved routers. This, however, is problematic for two main reasons: (1) compromising routers leads to HMAC keys leakage, and, more importantly, (2) if consumers provide routers with these keys, the former need to know the set of routers that their interests traverse before issuing them. Furthermore, since HMAC keys should only be shared among involved entities, i.e., Cr and all routers on $\mathsf{Path}(Cr, P)$, they must be distributed securely. Regardless of the distribution method used, this incurs extra overhead and complexity compared to simply including, in cleartext, signature verification (public) keys in content objects.

Finally, consider the following scenario where two routers R_1 and R_2 cache content object $CO[N']$ which is under IBAC protection. Assume that consumer Cr requests $CO[N']$ by sending an interest $I[N']$ with valid authorization information that includes r and t. Assume that $I[N']$ is satisfied from R_1's cache. At the same time, Adv, an eavesdropper between Cr and R_1, records $I[N']$. In this case, Adv can replay $I[N']$ to R_2 and receive $CO[N']$ from the cache since routers do not synchronize stored nonces. Therefore, there is no way for R_2 to know that r and t were already used at R_1. One way of solving this problem is to have routers share used nonces lists for each content under IBAC they serve from cache. For this method to be effective, such nonces lists need to be securely shared with every single router in the network. This might not be feasible in large networks such as the Internet. Another approach is to have more accurate synchronized clocks allowing a smaller time window for the aforementioned attack to be carried out.

6.2 Authorized Content-Key Binding Rule

Although the aforementioned method for generating authorization information mitigates replay attacks, it also raises several questions. Firstly, how does a router efficiently verify the signature in interest `Payload` fields? Secondly, and perhaps more importantly, if a router is able to *obtain* the key(s) necessary to verify this signature, how can the router be sure that such key(s) can be trusted?

To address these problems we propose a mutual trust framework for authorized disclosure. Ghali et al. [9] first studied the problem of trust in NDN, and ICNs in general, as a means of preventing content poisoning attacks [8, 6]. Even if routers can verify content signatures before replying from their cache, it does not mean that said content is actually *authentic*. Ghali et al. observed that this verification process requires insight about trust in public keys (used in verification) that is only known to applications. Consequently, this requires that all interests must either supply (1) the hash of the public key used to verify the signature, or (2) the hash of the requested content. In effect, the interest reflects the trust context of the issuing consumer in a form enforceable at the network layer. This framework can be viewed as one-way trust of content by routers. We extend this framework to allow producers to distribute information about authorized consumers, which can also be enforceable at the network layer. This allows routers to make trust decisions about individual interests.

Algorithm 1 InterestGeneration

```
1: INPUT: routable_prefix, $N$, $k_{\mathbb{G}_i}$, $pk^s_{\mathbb{G}_i}$, $sk^s_{\mathbb{G}_i}$, $\kappa$
2: $\mathsf{ID}_{\mathbb{G}_i} \leftarrow \mathsf{H}(k_{\mathbb{G}_i})$
3: $N' \leftarrow$ /routable_prefix/$f(k_{\mathbb{G}_i}, \mathsf{Suffix}(N, \text{routable\_prefix}))$
4: $r \xleftarrow{\$} \{0,1\}^\kappa$
5: $t \leftarrow \mathsf{CurrentTime}()$
6: $\sigma \leftarrow \mathsf{Sign}_{sk^s_{\mathbb{G}_i}}(N'||\mathsf{ID}_{\mathbb{G}_i}||r||t)$
7: Payload := $(\mathsf{ID}_{\mathbb{G}_i}, r, t, \sigma)$
8: return $I[N'] := (N', \texttt{Payload})$
```

Algorithm 2 ContentObjectGeneration

```
1: INPUT: $I[N'] := (\text{routable\_prefix}, N', \texttt{Payload})$
2: $(\mathsf{ID}_{\mathbb{G}_i}, r, t, \sigma) :=$ Payload
3: $pk^s_{\mathbb{G}_i} \leftarrow \mathsf{LoopupVerificationKeyForID}(\mathsf{ID}_{\mathbb{G}_i})$
4: if $\mathsf{Verify}_{pk^s_{\mathbb{G}_i}}(\sigma)$ then
5:     $k^e_{\mathbb{G}_i} \leftarrow \mathsf{LookupDecryptionKeyForID}(\mathsf{ID}_{\mathbb{G}_i})$
6:     $N \leftarrow \mathsf{Dec}(k^e_{\mathbb{G}_i}, \mathsf{Suffix}(N', \text{routable\_prefix}))$
7:     data $\leftarrow \mathsf{RetrieveContent}(N)$
8:     return $CO[N'] := (N', \text{data}, pk^s_{\mathbb{G}_i})$
9: else
10:    Drop $I[N']$
11: end if
```

Recall that in order for routers to verify which interests are authorized to access cached content protected under IBAC, the signature in `Payload` must be verified. To achieve this, producers should include the appropriate verification key with each IBAC-protected content object. To better understand this, assume the following scenario. Consumer $Cr \in \mathbb{G}_i$, for $\mathbb{G}_i \subset \mathbb{U}(N)$, requests content with name N by issuing an interest with obfuscated name N', and $\mathsf{ID}_{\mathbb{G}_i}$, r, t and σ in `Payload` as described in Section 6.1. Assume that the matching content is not cached anywhere in the network. Once this interest reaches the producer P, the latter verifies σ and replies with the content that also includes the verifying key $pk^s_{\mathbb{G}_i}$.[12] Router R will then cache $pk^s_{\mathbb{G}_i}$ along with the content itself. Once another interest for N' is received, R uses the cached $pk^s_{\mathbb{G}_i}$ to verify σ and returns the corresponding cached content object.

We formalize this in the following policy, denoted as the Authorized Content-Key Binding (ACKB) rule:

ACKB: Cached content protected under IBAC must reflect the verification key associated with the authorization policy.

The protocol for IBAC-protected content retrieval relies on this rule. Algorithms 1 and 2 outline the interest and content object generation procedures. Note that the function $\mathsf{Suffix}(N, \text{routable\_prefix})$ returns all name components

[12]The content object signature must also be computed over $pk^s_{\mathbb{G}_i}$ to bind it to the message.

Algorithm 3 RouterAuthorizationCheck

```
1:  INPUT: $I[N']$, cached $CO[N']$, $\mathbb{B}$
2:  $(\mathsf{ID}_{\mathbb{G}_i}, r, t, \sigma) :=$ Payload
3:  $(N', \cdot, pk^s_{\mathbb{G}_i}) := CO[N']$
4:  if $\mathbb{B}[N']$ contains $r$ then
5:      Drop $I[N']$; return Fail
6:  else
7:      if Timestamp $t$ is invalid then
8:          Drop $I[N']$; return Fail
9:      else
10:         if $\mathsf{Verify}_{pk^s_{\mathbb{G}_i}}(\sigma)$ then
11:             $\mathbb{B}[N'] := \mathbb{B}[N'] \cup r$
12:             return Pass
13:         else
14:             Drop $I[N']$; return Fail
15:         end if
16:     end if
17: end if
```

of N except the ones included in routable_prefix.[13] Also, the router verification procedure is outlined in Algorithm 3. If this procedure returns `Pass`, then the content object found in the cache is forwarded downstream to the associated interface. Note that Algorithms 1, 2, and 3 use obfuscation key $k_{\mathbb{G}_i}$ and signing key pairs $(pk^s_{\mathbb{G}_i}, sk^s_{\mathbb{G}_i})$. For completeness, a complete sequence diagram showing multiple interest-content exchanges is shown in Figure 1. Both consumers belong to the same access group, i.e., $Cr_1, Cr_2 \in \mathbb{G}_i$.

In [7], we show that this mutual trust framework for authorized disclosure enables IBAC with stronger security guarantees in the presence of replay attacks.

6.3 Serving Content to Multiple Access Groups

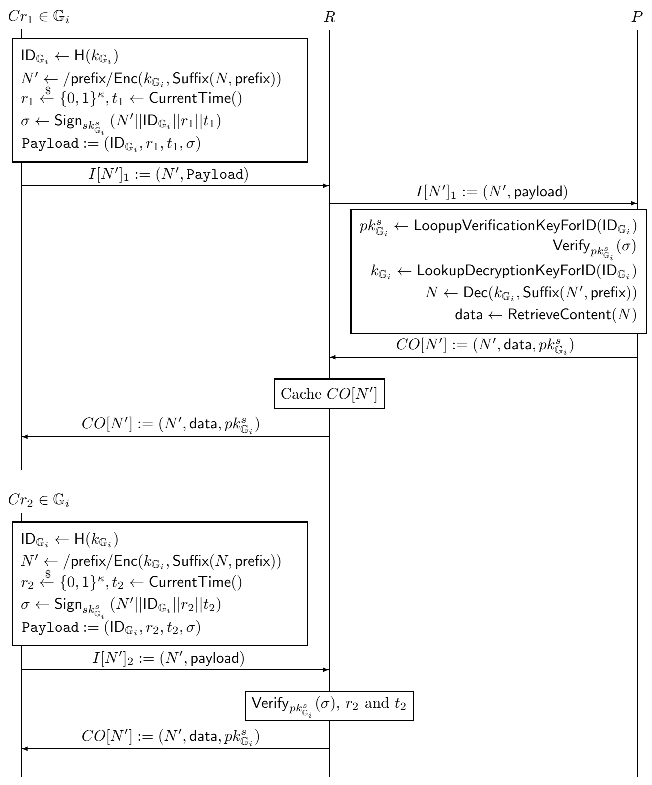
One problem with encryption-based name obfuscation occurs when a content object with name N is accessible by different groups. According to Algorithms 1 and 2, the obfuscated name N' contains a suffix encrypted with keys associated with each access group. Therefore, a single content object might have several names depending on the number of groups authorized to access it. Since routers employs exact matching for cache lookup[14], several copies of the same content could possibly be cached.

To solve this problem, content objects should have the exact same name regardless of access control groups permitted access. This can be achieved using the hash-based name obfuscation function described in Section 5.2. However, cached content needs to contain *every* authorization signature verification key that could be used to access said content. In other words, producers need to provide all possible public keys that can be used to access the content under IBAC protection. Consider the following scenario: a content object $CO[N]$ is accessible by two access groups $\mathbb{G}_i$ and $\mathbb{G}_j$. In this case, the producer needs to provide both $pk^s_{\mathbb{G}_i}$ and $pk^s_{\mathbb{G}_j}$ with $CO[N']$, i.e.,

$$CO[N'] := (N', \mathsf{data}, pk^s_{\mathbb{G}_i}, pk^s_{\mathbb{G}_j})$$

Whenever a router R caching $CO[N']$ receives an interest issued by a consumer in any of the authorized access groups, R uses the group identity included in the `Payload` field to determine σ's verification key.

[13]For instance, Suffix(`/edu/uci/ics/home.html`, `/edu/uci/`) would return `ics/home.html`.
[14]In CCN, not in NDN.



$Cr_1 \in \mathbb{G}_i$
R
P
$\mathsf{ID}_{\mathbb{G}_i} \leftarrow \mathsf{H}(k_{\mathbb{G}_i})$
$N' \leftarrow /\mathsf{prefix}/\mathsf{Enc}(k_{\mathbb{G}_i}, \mathsf{Suffix}(N, \mathsf{prefix}))$
$r_1 \xleftarrow{\$} \{0,1\}^\kappa, t_1 \leftarrow \mathsf{CurrentTime}()$
$\sigma \leftarrow \mathsf{Sign}_{sk^s_{\mathbb{G}_i}}(N' \| \mathsf{ID}_{\mathbb{G}_i} \| r_1 \| t_1)$
$\mathtt{Payload} := (\mathsf{ID}_{\mathbb{G}_i}, r_1, t_1, \sigma)$
$I[N']_1 := (N', \mathtt{Payload})$
$I[N']_1 := (N', \mathsf{payload})$
$pk^s_{\mathbb{G}_i} \leftarrow \mathsf{LoopupVerificationKeyForID}(\mathsf{ID}_{\mathbb{G}_i})$
$\mathsf{Verify}_{pk^s_{\mathbb{G}_i}}(\sigma)$
$k_{\mathbb{G}_i} \leftarrow \mathsf{LookupDecryptionKeyForID}(\mathsf{ID}_{\mathbb{G}_i})$
$N \leftarrow \mathsf{Dec}(k_{\mathbb{G}_i}, \mathsf{Suffix}(N', \mathsf{prefix}))$
$\mathsf{data} \leftarrow \mathsf{RetrieveContent}(N)$
$CO[N'] := (N', \mathsf{data}, pk^s_{\mathbb{G}_i})$
Cache $CO[N']$
$CO[N'] := (N', \mathsf{data}, pk^s_{\mathbb{G}_i})$
$Cr_2 \in \mathbb{G}_i$
$\mathsf{ID}_{\mathbb{G}_i} \leftarrow \mathsf{H}(k_{\mathbb{G}_i})$
$N' \leftarrow /\mathsf{prefix}/\mathsf{Enc}(k_{\mathbb{G}_i}, \mathsf{Suffix}(N, \mathsf{prefix}))$
$r_2 \xleftarrow{\$} \{0,1\}^\kappa, t_2 \leftarrow \mathsf{CurrentTime}()$
$\sigma \leftarrow \mathsf{Sign}_{sk^s_{\mathbb{G}_i}}(N' \| \mathsf{ID}_{\mathbb{G}_i} \| r_2 \| t_2)$
$\mathtt{Payload} := (\mathsf{ID}_{\mathbb{G}_i}, r_2, t_2, \sigma)$
$I[N']_2 := (N', \mathsf{payload})$
$\mathsf{Verify}_{pk^s_{\mathbb{G}_i}}(\sigma)$, r_2 and t_2
$CO[N'] := (N', \mathsf{data}, pk^s_{\mathbb{G}_i})$

Figure 1: Consumer and producer exchanges for IBAC-protected content.

Note that content object sizes might increase significantly depending on how many groups are allowed access. We do not discuss this issue further, since the trade-off between having multiple cached versions of the same content and having longer content objects carrying all verification keys is ultimately the application's decision.

6.4 IBAC Variations

We do not claim that any of the IBAC variations discussed above is superior to another. Each has its own strengths and weaknesses. However, to help guide the decision about which variation to use, we make the following claims based on the application needs and assumptions. Note that some claims provide privacy as well as access control.

1. If *replay attacks* are not a concern, then consumers only need to use a name obfuscation function and include their group identity in the `Payload`.

2. If *replay attacks* are plausible and *name privacy* is a concern, then name obfuscation must be used and authorization information, as described in Section 6.1, must be included in interest `Payload` fields.

3. If *replay attacks* are plausible but *name privacy* is not a concern, then only authorization information is sufficient.

Claim 3 might seem counterintuitive with the idea of IBAC. Recall, however, that router authorization checks prevent unauthorized consumers from retrieving cached content under IBAC protection. Even if content name is not obfuscated, Adv cannot forge `Payload` authorization information, and therefore cannot violate IBAC protection guarantees.

6.5 Revocation

Generally speaking, revocation is a challenge in all access control schemes involving secrets shared among group members. Recall that all consumers belonging to the same access control group in IBAC share the same obfuscation keys. If one of them leaves the group[15], the producer will have to create a new key and distribute it to all remaining authorized consumers. We will not discuss this issue further since we believe it is not part of the core access control *protocol*.

Moreover, in-network caching can cause IBAC content to be accessed by revoked consumers. Assume content $CO[N]$ is under access control and has a cached version in router R. Assume consumer Cr, connected (directly or indirectly) to R, is authorized to access $CO[N]$. However, while $CO[N]$ is cached, Cr's access is revoked. At the same time, the latter sends an interest requesting $CO[N]$. In this case, R will grant access and reply with $CO[N]$ from its cache. This is due to the fact that the cached version of $CO[N]$ is not updated with the correct authorization information (i.e., verification key(s)). However, this can be solved by setting the `ExpiryTime` field of $CO[N]$ to a value that reflects consumer revocation frequency.

Online revocation protocols, such as OCSP [20], would induce extra communication between R and P, which nearly defeats the purpose of the cache entirely. In this case, R would be better suited forwarding the interest upstream to P. Another option for the producer would be to distribute certificate revocation lists (CRLs) [4] with every fresh content. This, however, introduces further issues for routers and consumers. Firstly, routers would need to store CRLs and keep them updated frequently. Secondly, authorized consumers would need their own public and private key pair to compute σ. Finally, routers would need to perform additional verifications against the CLR. Overall, this approach suffers from increased storage, consumer management, computation, and bandwidth complexity.

7. ANALYSIS AND EVALUATION

In this section, we analyze the overhead induced by each variation of the proposed IBAC scheme.

7.1 Computational Overhead

We first focus on the computational overhead for routers and producers. This overhead is captured in terms of cryptographic and data structure operations, e.g., signature verification and hash table lookup costs.[16] Table 2 summarizes these results. To further understand the computational overhead, we compare two cases: (1) when routers perform authorization checks, and (2) when they do not. Let $\tau_{overhead} = \tau_{check} + \tau_{verify} + \tau_{update}$ be the overhead induced by the authorization check when routers receive interests, where τ_{check} is the time required to check for nonce duplication and timestamp staleness, τ_{verify} is the time to verify the `Payload` signature, and T_{update} is the time to update the nonce data collection. Since cache lookup and interest forwarding are performed regardless of whether or not routers perform authorization checks, we omit them from this equation. Similarly, τ_{check} and τ_{update} are negligible when compared to the cost of signature verification τ_{verify}; thus, they are also excluded.

A router incurs a computational cost of $\tau_{overhead}$ for every received interest requesting content under IBAC protection. Therefore, we quantify $\tau_{overhead}$ by measuring the time it takes to perform a single signature verification. We also experiment with batch verification techniques to better amortize the cost of signature verification across series of interests. While this naturally increases content retrieval latency as signatures are accumulated in case of batch verification, it reduces router computational overhead. Whether or not to use batch verification is up to the router's discretion. Furthermore, batch verification requires that IBAC-protected content objects for which interests are being verified cannot be evicted from the cache while the batch is collected. Table 3 shows the amount of improvement using a variety of signature verification algorithms. Note that, when modeling interest arrival rates using a Poisson distribution, both individual and batch signature verification incur nearly the same overhead in certain conditions, as we will show below.

Table 3: Individual and batch ElGamal signature verification times.

Key Size	Batch Size	Sig. Size	Indiv. Time	Batch Time	Improved
1024b	10	512KB	0.599s	0.322s	46%
1024b	10	8MB	0.888s	0.615s	30%
1024b	50	512KB	2.918s	1.579s	46%
1024b	50	8MB	4.315s	2.991s	30%
2048b	10	512KB	4.065s	2.207s	46%
2048b	10	8MB	4.104s	2.269s	45%
2048b	50	512KB	20.081s	11.029s	45%
2048b	50	8MB	21.301s	12.536s	41%
3072b	10	512KB	12.406s	6.789s	45%
3072b	10	8MB	12.804s	7.122s	44%
3072b	50	512KB	60.174s	32.877s	45%
3072b	50	8MB	64.347s	35.601s	45%

Denial of service (DoS) is an obvious concern if routers perform authorization checks (the interest rate decays to 0 in our experiments as the need for verification increases). Let λ be the rate of arrival interests for IBAC-protected content cached in router R, and let μ be the service rate for interests, i.e., the rate at which interests are processed (parsed, verified, etc.). If $\mu < \lambda$, then the router will be over encumbered with interests to process [10]. We envision that in legitimate scenarios without malicious entities generating interests with fake authorization information, only a small percentage δ of arrival interests will be requesting content under IBAC protection. To assess how susceptible routers are to DoS attacks induced by IBAC authorization checks, we empirically analyze the effect of δ on the interest service rate of a router. These service rates, which use different signature verification techniques – individual and batch – denoted μ_S and μ_B, respectively, are shown in Figure 2.

We assume that interests arrive at a base rate of $\lambda_1 = 40$ [3]; larger values for λ are provided to see at which point

[15]For instance, consumers not renewing their subscription for a certain service.

[16]It is assumed that the cost of any additional checks necessary to determine if an interest requires further IBAC processing is negligible. For example, this check be done using a simple flag in the interest. Thus, this overhead is omitted from our estimates.

Table 2: Overview of per-interest IBAC-induced computational overhead for routers and producers.

IBAC Variation		IBAC-induced Computation Overhead	
		Routers	**Producers**
Name Obfuscation	**Encryption**	None	One decryption
	Hash	None	One hash table lookup
Interest Signatures	**Encryption**	One signature verification, one nonce and timestamp verification	One decryption, one signature verification, Two hash table lookups (decryption key and signing key resolution)
	Hash	One signature verification, one nonce and timestamp verification, one hash table lookup (signing key resolution)	One signature verification, three hash table lookups (decryption key, signing key and name resolution)

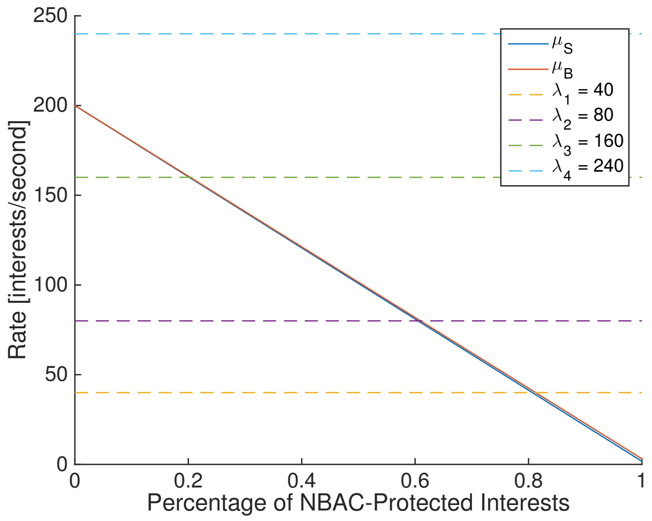


Figure 2: Interest service rates for various percentages of IBAC-protected interests.

$\mu < \lambda$ due to authorization checks. By the exponential property of the Poisson process, μ is calculated as follows:

$$\mu = \frac{1-\delta}{\tau_{process}} + \frac{\delta}{\tau_{process} + \tau_{verify}},$$

where $\tau_{process}$ represents interest processing time not including signature verification[17], and τ_{verify} is the time required to perform individual or batch signature verification. In our experiment, we assume a constant $\tau_{process} = 0.005$s and only vary τ_{verify}. To do so, we assume a key size of 1024b, batch size of 10, and signature size of 512KB. According to Table 3, this results in $\tau_{verify} = 0.599$s and $\tau_{verify} = 0.322$s for individual and batch verification, respectively. Our experiments show that the decay of μ as a function of δ is almost identical for both batch and verification techniques. This is due to the fact that only a small fraction of interests are affected by the verification step. Furthermore, our results show that $\mu > \lambda$ is true, i.e., the router servicing process is stable for reasonable interest arrival rates. Our experiments show that $\mu < \lambda$ when $\lambda = 160$ and $\delta \geq 0.2$. Moreover, when a Poisson process is assumed, both individual and batch signature verification perform similarly for small values of δ. However, batch signature verification prove to be advantageous in larger δ values. For instance, for $\delta = 0.2$, using batch verification provides less than 1% service rate improvement, where it provides 3% and 46% for δ values equal to 0.8 and 1, respectively.

[17] $\tau_{process} = 1/mu$ for interests not requesting IBAC-protected content.

7.2 Storage Overhead

Storage overhead varies from producer to router. If hash-based name obfuscation is used, producers incur the cost of maintaining a hash table to map obfuscated names to their original values. However, if content name contains variable name components, e.g., query string-like values in URIs, the hash table size can grow significantly since it has to contain all possible variations. Moreover, producers must bear the storage cost of IBAC access group keys if encryption-based obfuscation functions are used. Similarly, routers must bear the cost of storing variable-length tuples of key identities $\mathsf{ID}_{\mathbb{G}_i}$ and the actual verification keys $pk^s_{\mathbb{G}_i}$, along with a theoretically unbounded collection of nonces for each IBAC-protected content. Moreover, these storage costs are paid for every unique producer that generates IBAC content. Clearly, router finite storage capacities can thus be abused to mount DoS attacks.

7.3 Bandwidth Overhead

In terms of bandwidth overhead, each interest and content object is expanded to include additional authorization information, e.g., interest payloads with authorization information and content objects with authorization keys. Interests without authorization payloads will only increase (or decrease) by the expansion factor of the obfuscated name. If authorization payloads are included, then interest messages will grow by $|r| + |t| + |\sigma| + |\mathsf{ID}_{\mathbb{G}}|$, where $|r| = \kappa$. Content object $CO[N]$ grows with length $\sum_{i=1}^{L} |pk^s_{\mathbb{G}_i}|$, where L is the number of access groups allowed to access $CO[N]$ and $|pk^s_{\mathbb{G}_i}|$ is the public key size associated with group $\mathbb{G}_i$.

Note that, for interests, this increased overhead may cause them to grow beyond the MTU size, which would induce interest fragmentation. However, given that names may themselves be unbounded, this type fragmentation seems unavoidable in certain circumstances.

8. CONCLUSION

We studied the problem of access control in CCN. We proposed an Interest-Based Access Control (IBAC) scheme that supports hash- and encryption-based name obfuscation. We addressed the problem of replay attacks by formulating a mutual trust framework between producers and consumers – enforced in the network-layer – that enables routers to perform authorization checks before satisfying interests from cache. We assessed the computational, storage, and bandwidth overhead induced by each variant of the proposed IBAC scheme.

9. REFERENCES

[1] Microsoft PlayReady. http://www.microsoft.com/playready/.

[2] T. Berners-Lee, R. Fielding, and L. Masinter. RFC 3986: Uniform resource identifier (URI): Generic syntax. 2005.

[3] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino. Modeling data transfer in content-centric networking. In *ITC*, 2011.

[4] D. Cooper. RFC 3280: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. 2008.

[5] N. Fotiou, G. F. Marias, and G. C. Polyzos. Access control enforcement delegation for information-centric networking architectures. In *ICN Workshop*, 2012.

[6] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. DoS DDoS in named-data networking. In *ICCCN*, 2013.

[7] C. Ghali, M. A. Schlosberg, G. Tsudik, and C. A. Wood. Interest-based access control for content centric networks (extended version). *arXiv*, 2015.

[8] C. Ghali, G. Tsudik, and E. Uzun. Needle in a haystack: Mitigating content poisoning in named-data networking. In *NDSS SENT Workshop*, 2014.

[9] C. Ghali, G. Tsudik, and E. Uzun. Network-layer trust in named-data networking. *CCR*, 2014.

[10] D. Gross. *Fundamentals of queueing theory*. John Wiley & Sons, 2008.

[11] M. Ion, J. Zhang, and E. M. Schooler. Toward content-centric privacy in ICN: Attribute-based encryption and routing. In *ICN*, 2013.

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

[13] S. Jahid, P. Mittal, and N. Borisov. EASiER: Encryption-based access control in social networks with efficient revocation. In *ASIACCS*, 2011.

[14] S. Kamara and K. Lauter. Cryptographic cloud storage. In *FC*, 2010.

[15] J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC Press, 2014.

[16] H. Krawczyk, R. Canetti, and M. Bellare. RFC 2104: HMAC: Keyed-hashing for message authentication. 1997.

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

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

[19] M. Mosko, I. Solis, and E. Uzun. CCN 1.0 protocol architecture.

[20] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. RFC 2560: Online certificate status protocol - OCSP. 1999.

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

[22] G. Wang, Q. Liu, and J. Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *CCS*, 2010.

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

[24] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM*, 2010.

[25] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang, et al. Named data networking. *CCR*, 2014.

[26] L. Zhou, V. Varadharajan, and M. Hitchens. Achieving secure role-based access control on encrypted data in cloud storage. *Transactions on Information Forensics and Security*, 2013.