A consumer-driven access control approach to censorship circumvention in content-centric networking

Jun Kurihara, Kenji Yokota and Atsushi Tagami

KDDI R&D Laboratories, Inc.

ACM ICN 2016
Kyoto, Japan, Sep. 28, 2016
Outline of my talk

1. Introduction
2. Censorship circumvention in CCN
3. Basics of consumer-driven access control approach
4. Enhancement using manifest and nameless object
5. Conclusion
Introduction
Censorship: A serious problem in networking

Censorship in a network:
Monitoring network messages, checking ‘what is requested’, and dropping messages in the blacklist by a certain authority.
Censorship is easily enforceable in CCN

Content data itself can be encrypted in a certain AC, but interest name is not. Explicitly-given and semantic name in CCN made censorship trivial.

- Capture and analyze interests; and
- Drop any interests by checking only their names “democracy”
Censorship circumvention in CCN
Two types of countermeasures in CCN

• Tor-like scheme
  • Multi-layered encryption at anonymizing routers
  • Significant overhead and delay

• Proxy-based scheme
  • Establishing anonymized channel between proxy and consumer
  • Simpler and faster than Tor-like scheme


Proxy-based approach

Our scheme is basically categorized as a proxy–based scheme

Anonymized interest
(/<routable prefix>/ + encrypted name)

encrypt!

Communication via encrypted name

trusted proxy

decrypt!

interest

plaintext name

domain: /kddi

/kddi/democracy.mpg
Cache recycling problem of proxy-based approaches

Anonymized communication is established between each consumer and a proxy under distinct encryption key.

Anonymized communication channel

Consumer A

Consumer B

The same content is queried via different names by different users

Cached content never be recycled
Basics of consumer-driven access control approach
System model

- Entity: CCN basic parties + cache enablers $E_i$ + anonymizer $A$ + attacker

- Content names follow a conventional (ICN) hierarchical naming scheme like URL (e.g., /kddi/demo/video.mpg).
Attacker definitions

We consider two types of attackers.

**Passive Attacker**
- Capture/analyze interests
- Learn “what is requested” and “who is requesting”;
- Drop/filter interests

**Active Attacker**
- Capture/analyze interests
- Modify interests
- Masquerade as legitimate consumers

*Passive ⊃ Active*
Key elements of our approach

[Against passive attacker]

(1) Encryption-based access control to interest names for cache enablers and anonymizer

[Against passive/active attacker]

(2) Authentication and decryption with hidden consumer ID at cache enablers and anonymizer
Access control:
A technique used to *regulate who or what can view raw/original data* in a computing environment.

Encryption-based access control:
Data is encrypted in such a way that *only authorized users are allowed to decrypt the encrypted data and obtain the raw data.*

Assigned decryption keys are identified as *access rights*
(1) Encryption-based access control to names: Overview of the approach

Consumer grants access rights to original interest names to cache enablers $E_i$ and anonymizer $A$ via the encryption-based access control.

Assign key for $E_1$

Assign key for $E_2$

Assign key for $A$

domain: /kddi
Consumers encrypt interest names in such a way that pre-authorized $E_i$ and $A$ can decrypt them and obtain original names.

**[Processing incoming interest at $E_i$]**

- Anonymized interest (/routable prefix/ + encrypted name)
- (1) Decrypt (/kddi/democracy.mpg)
- (2) CS search with original name
- (3) Respond by encrypted name

*** illustrated only the case of cache hit for simplicity. ***
[Processing incoming content at $E_i$] (simply the dual of interest case)

(1) Decrypt /kddi/democracy.mpg

(2) Cache with original name for recycle

[Key observation]

Access control to interest names ↔ Access control to cache-recycling opportunities

*** omitted the process of PIT entry consumption for simplicity. ***
(2) Authentication and decryption with hidden ID: Preliminary

[Observations]

\( E_i \) and \( A \) must learn the consumer ID from an interest to find a consumer specific key(s) for name decryption and interest authentication via HMAC/signature

\[ \downarrow \]

Consumer ID itself leaks the consumer information to attackers

[Requirements]

- Consumer ID must be included and hidden in interests
- Only cache enablers and anonymizer learn the ID from an interest for decryption and authentication
Anonymizer uses a public key broadcast encryption for hiding IDs in interests.

- Decryption keys are assigned to cache enablers
- Public (encryption) key is published.
Consumer generates the anonymizing interest from the encrypted name as:

Broadcast public key from A

Encrypted name

(Consumer ID)

Encrypted ID

HMAC

HMAC generation by name encryption key
$E_i$ and $A$ authenticate and generate the incoming interest as:

$E_i$ and $A$ authenticate and generate the incoming interest as:

$\text{(Consumer ID)}$
Advantage and disadvantage

[Security for passive attacker]
- No leakage about content name (what)
- No leakage about consumer identity (who)

[Security for active attacker]
- Interest modification can be detected
In-network caching can be fully leveraged at cache enablers $E_i$’s and anonymizer $A$

More beneficial as # of $E_i$’s increases.

Cryptographic operations (access control and authentication) at $E_i$ and $A$ may involve serious computational cost.

More serious overhead as # of $E_i$’s increase.

Trade-off between cache recycling opportunity and overhead
This problem is solved by combining our approach with manifest and nameless objects.

We minimize the overhead with maintaining the security and maximizing the benefit of in-network caching.
Enhancement using manifest and nameless object
**Preliminary: Manifest and nameless objects in CCNx**

**Manifest:** Content object providing a list of content objects (names and hashes)

**Manifest structure**

<table>
<thead>
<tr>
<th>Names</th>
<th>Hashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>/kddi/democracy.mpg/1</td>
<td>0xABCD</td>
</tr>
<tr>
<td>/kddi/democracy.mpg/2</td>
<td>0x1234</td>
</tr>
<tr>
<td>/kddi/democracy.mpg/3</td>
<td>0xA1B2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Guarantee of integrity and unforgeability

**Manifest-based content retrieval:**
Consumer first obtain and parse manifest, then retrieve listed content objects.

**Listed items can be authenticated only by lightweight hash verification.**
**Nameless object: a variant of content object**

- Content object payload is **decoupled with name**.
- Queried by **arbitrary-given but correctly-routable** name + its hash value.

<table>
<thead>
<tr>
<th>Name</th>
<th>Original</th>
<th>Replica 1</th>
<th>Replica 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/kddi/democracy.mpg/1</code></td>
<td>hash = 0x1234ABCD</td>
<td><code>/kyoto/movie.mpg/1</code></td>
<td><code>/anonymized/v.mpg/1</code></td>
</tr>
</tbody>
</table>

- Used for interest routing
- Used for CS/PIT search

**Content replica redirection** can be easily realized.

**Decoupled from name**

Note: **Consumer needs to first retrieve a manifest** in order to learn routable names and hashes for nameless objects.
Maximizing benefit of in-network caching with minimizing computational overhead

**Assumption**: Desired content objects are encrypted under appropriate access control (like CCN-AC*), and attacker does not know their hashes.

**Assumption**: Desired content objects are **nameless objects** and hosted at a certain *consumer-reachable* replication server with meaningful (uncensored) names.

**Important observation**: The name of replicated content object itself is semantically meaningless.

--> **Nameless objects are never filtered based on name.**

The 2-phased strategy of enhancement:

• **[Phase 1]**
  Manifest and non-replicated extra information (e.g., decryption keys) are retrieved by consumer-driven access control approach.

  -> Our secure but heavy approach is used only for manifest + α

• **[Phase 2]**
  Replicated nameless content objects are simply queried in the standard manner of content retrieval.

  -> Large number of objects are never be filtered even in the standard manner.
Example of minimization of computational cost in flow:

- **Phase 1**
  - anonymized interest for a manifest
  - interests for listed nameless objects

- **Phase 2**
  - listed nameless objects
  - No cryptographic operations at intermediate nodes in phase 2!

- Replicating nameless objects
- Anonymizer a.k.a. replication server
- Publisher

Cryptographic operations on interest name
Example of cache recycling opportunities in flow:

Phase 1
- anonymized interest for a manifest
- interests for listed nameless objects

Phase 2
- interest for a manifest
- listed nameless objects

Every node has recycling opportunity in phase 2!

- replicating nameless objects
- anonymizer a.k.a. replication server
- publisher

Opportunity to respond from cache
Conclusion
Conclusion and future work

• In this talk:
  • We introduced a proxy-based censorship circumvention approach enabling in-network caching.
    • Consumer-driven access control to interest names
    • Authentication and decryption with hidden consumer ID
  • We enhanced the approach by using manifest and nameless objects
    • Maximizing the cache recycling opportunity
    • Minimizing the overhead of cryptographic computation at intermediate nodes

• Future work:
  • Implementation and performance evaluation in realistic environment with specific settings.
  • etc.
Thank you!

Comment and question...?

e-mail: kurihara@ieee.org