2018 IEEE 38th International Conference on Distributed Computing Systems

TACTIC: Tag-based Access ConTrol Framework for the Information-Centric Wireless Edge Networks

Reza Tourani, Ray Stubbs, Satyajayant Misra*
New Mexico State University
{rtourani, oril, misra}@cs.nmsu.edu

Abstract—**Pervasive content caching is one of the information-centric networking (ICN) fundamentals. Although advantageous, pervasive caching introduces new challenges. In particular, the high possibility of content providers losing control over their published contents, which clients can access without authenticating themselves. The approaches that constitute the state-of-the-art in access control either have high computation overhead or require an always-online authentication server, thus suffering in terms of scalability for large number of end devices.**

In this paper, we propose TACTIC, a lightweight access control mechanism for the ICN wireless edge, which allows legitimate clients to utilize the cached content without per-request authentication at the providers. TACTIC delegates the authentication and authorization tasks to the (semi-trusted) routers in an ISP's network to eliminate the need for an always-online authentication server. It prevents delivery of the encrypted content to unauthorized users; a bandwidth-wasteful practice, which may lead to Distributed Denial of Service (DDoS) attack. Experimental results demonstrate the scalability and effectiveness of TACTIC in providing low-overhead access to legitimate clients while preventing malicious users' access.

Index Terms—**Information-centric networking, access control, authentication, authorization, wireless edge, IoT.**

1. Introduction

The number of wireless devices and connections are growing faster than the world population and Internet users. As per Cisco, major drivers of this growth are smartphones and various machine to machine (M2M) applications, such as smart meters, asset tracking, and video surveillance [1]. This immense number of mobile devices and connections, which will grow from 8 billion to 11 billion by 2021, calls for a scalable architecture that can accommodate the corresponding large traffic volume from heterogeneous devices [2].

Information-centric networking (ICN) has been proposed for shifting the existing TCP/IP "host-centric" model to a connection-less "content-centric" paradigm to improve the clients quality of experience (QoE) and overall network quality of service (QoS). In a TCP/IP network, a mobile user has a unique connection between itself and a server (data source). The data transmitted by the network cannot be reused for satisfying requests from other clients. Further, if the user moves, the connection has to be re-initiated. In ICN, however, the mobile client seamlessly resumes its content retrieval when it connects to its new base station (or access point) and caching and content reuse are ubiquitous, thus helping speed up content retrieval. Th characteristics help reduce communication overhead, latency, and device energy consumption.

Despite several advantages, such as pervasive caching, built-in security, and lower content retrieval latency, ICN introduces a few challenges that need attention. Efficient access control enforcement is among these challenges; resulting from pervasive content caching. A content object, when published by its publisher, can be cached at every node in the network allowing subsequent requests for the content to be fulfilled from these in-network caches. However, these cache hits prevent the content providers from receiving the requests, and hence authenticating and authorizing the requester. This calls for efficient mechanisms that provide strong authentication and authorization to prevent unauthorized entities with insufficient privileges from accessing cached content.

Motivation: The contemporary access control approaches, in which a client authenticates herself to the provider for obtaining the content decryption key, are not desirable due to their host-centric communication model and their dependency on an always-online authentication server. If used in ICN, these approaches prevent a client that can obtain the encrypted cached content from the network from decrypting and consuming it, particularly if the authentication server is not available. More importantly, even if an authentication server is always-online, after each client revocation the providers have to re-encrypt and re-disseminate the content into the network, to prevent revoked users from accessing the cached content. A practice that incurs computation and communication overheads.

As a solution, a class of ICN-based access control mechanisms [3]–[5] suggests the delegation of the authorization process to the end clients themselves. That is, in these mechanisms all users can retrieve the content from the network. However, only legitimate clients with sufficient authorization information (provided during a prior authorization process) can decrypt and consume the content. Despite the feasibility, such mechanisms are prone to wasting of network bandwidth and potential network Distributed Denial of Service (DDoS) attack by unauthenticated or revoked users.

To overcome the aforementioned shortcomings, in this paper we propose *TACTIC*, an access control mechanism for the ICN wireless edge. In TACTIC, providers delegate access

* This work was supported in part by the US National Science Foundation Grants 1248109, 1345232, and 1719342. The information reported here does not reflect the position or the policy of the federal government.

control enforcement to the routers, allowing cache utilization and promoting content availability with negligible computation and communication overhead. A client registers herself at a content provider and receives an authentication tag, which she includes in her requests to prove her access privilege. A router, on receiving such a request, validates the integrity and provenance of the tag and returns the content if the tag is valid. TACTIC is designed to be relevant for a wide range of clients, which will make up tomorrow's mobile edge devices (e.g., cars, smartphones, and other IoT/CPS devices).

In a nutshell, our **contributions** include: ***(i)*** Detailed design of ***TACTIC***, an efficient ICN-based access control mechanism, in which authorization is delegated to the network entities to eliminate the need for an always online authorization server or client-end authorization. ***(ii)*** Discussions on TACTIC's design and implementation issues in the NDN architecture. ***(iii)*** Comparison of TACTIC with the state-of-the-art in ICN access control and discussion on its efficiency for constrained devices. ***(iv)*** Implementation of TACTIC in the ndnSIM simulator, with simulation results validating its scalability.

The paper is organized as follows. Section 2 presents the related work. In Section 3, we explain our models and assumptions. Section 4 presents TACTIC's features and its brief overview. In Section 5, we elaborate on TACTIC's implementation in an ICN architecture. We discuss the security implications of our mechanism in Section 6 and present our simulation results in Section 7. Finally, we draw our conclusion and present our future work in Section 8.

2. Related Work

In this section, we first introduce *Named-Data Networking* (NDN), one of the popular ICN architectures, which is the ICN architecture we use in this paper, and then review the state-of-the-art in ICN access control.

In NDN, network entities are equipped with a *Forwarding Information Base (FIB)*, *Pending Interest Table (PIT)*, and a *Content Store (CS)*. Upon receiving an *Interest* (an NDN named-request) from a client for some content, the receiving router performs a CS (cache) lookup and returns the content if it is available. Otherwise, the router performs a PIT lookup to check whether there is an in-flight Interest for the content or not. If a PIT entry exists, the router aggregates the received Interest, by adding its incoming face (interface) to the existing PIT entry, and drops it. Otherwise, the router creates a new PIT entry for the Interest, consults the FIB to select an outgoing face, and forwards the Interest towards the content provider over the chosen face. NDN employs reverse path forwarding for the content packets; a router receiving a content packet searches the content name in its PIT to find the face(s) over which it needs to forward that content packet.

Existing research in ICN access control management include approaches using encryption-based, attribute-based, identity-based, and proxy re-encryption techniques. In what follows, due to limited space we review the best approaches, and refer the interested readers to a survey on ICN access control [6].

Misra *et al.* [3], [7] proposed a broadcast encryption-based access control framework, which leverages Shamir's secret sharing. Although it has effective client revocation, this framework incurs a non-trivial communication and computation overhead every time a client requests a content. Chen *et al.* proposed a probabilistic access control mechanism in [8], in which routers use Bloom filters for storing a valid client's public key that allows early request filtration. The main drawbacks of this mechanism are the need for an always-online publisher and the impact of Bloom filter false positives. Kurihara *et al.* [9] proposed a similar approach in which the content encryption key is encrypted and disseminated into the network. A client needs to contact the provider for acquiring the content decryption key– a mechanism requiring an always-online authentication server along with its proposed lazy client revocation.

Attribute-based access control has been widely used by the community [4], [10]–[13]. Despite the similarity of the proposed mechanisms, only the work in [10], [11] considered client revocation. Da Silva [10] suggested a per revocation key update in contrast to the system re-key proposed by Hamdane [11]. The majority of the proposed mechanisms require additional infrastructure for key generation and distribution, and also result in significant communication overheads. While most of these mechanisms offloaded the computational cost to either the provider or the consumer, the mechanism proposed by Da Silva *et al.* [10] delegated the authentication task to the intermediate routers.

Wood *et al.* [14] proposed an identity-based access control scheme in which a provider encrypts the content decryption key with the client's identity and sends it to the client on successful authentication. However, the provider-based authentication requires the provider to be always-online. Mangili *et al.* [5] designed ConfTrack-CCN to enforce access control with trackability. Breaking a content into partitions and further fragments allows two layers of encryption, where the first layer key is embedded in the second and the second layer key is generated by clients upon obtaining the provider's secret. For client revocation, the provider encrypts the first layer key with a new second layer key and updates it in the network. The drawback of this mechanism is the complexity of associating keys to fragments.

Tan *et al.* [15] proposed a copyright protection scheme in the form of an access control mechanism. They break a content object into two partitions; the bigger partition is cached in the network while the smaller partition remains at the producer for client authentication. Li *et al.* [16] proposed a token-based access control in which providers publish public and private tokens to unauthorized and authorized clients, respectively. A client's request retrieves the content if the provider successfully validates the client's token. The main drawback of these two schemes were the dependency on the always-online authentication server.

Another set of solutions [8], [10] require the network (routers) to enforce access control and authenticate clients. The advantages of these approaches include: (i) do not require an always-online server at the provider and (ii) access control can be distributed across the network and is resilient. *However, the fact that the intermediate routers have to perform cryptographic operations undermines the practicality of these approaches. In this paper, we address this inefficiency and propose a scalable in-network access control mechanism with negligible computation overhead on the network entities.* In our mechanism, clients' tags, obtained from the providers are used by the intermediate entities for enforcing the content access. TACTIC eliminates the content re-encryption requirements of other approaches and relies on a few signature verifications and constant time Bloom filter operations.

3. Models and Assumptions

In this section, we present our system model, security assumptions, and threat model.

A. System Model

In this paper, we consider an ISP network with wireless devices at its edge. The network is hierarchical including wireless and mobile devices ($\mathcal{U}$) at the lowest layer, which are connected to the edge routers ($\mathcal{R}_E$) via wireless access points (APs), the ISP core routers ($\mathcal{R}_C$), and the content providers ($\mathcal{P}$) on top of the hierarchy (refer to Fig. 1). Routers are either edge routers ($\mathcal{R}_E$) or core routers ($\mathcal{R}_C$), where $\mathcal{R}_E \cup \mathcal{R}_C = \mathcal{R}$. For a given content, core routers are either *content routers ($\mathcal{R}_C^c \subseteq \mathcal{R}_C$)*, if the the content has been cached, or *intermediate routers ($\mathcal{R}_C^i \subseteq \mathcal{R}_C$)*, otherwise. We envision a Bloom filter (*BF*) for each router ($r \in \mathcal{R}$). Bloom filter is a probabilistic data structure that uses hash functions for testing the membership of an element in a set. Bloom filters have false positive when a query for an element returns true while the element is not a member of the set (hash collisions).

In our model, a content provider p ($p \in \mathcal{P}$) defines different access levels (AL) for its contents. A content access level is included in the content's packets and signed by the provider to guarantee its integrity and provenance. We use the named-data networking (NDN) architecture as the ICN architecture.

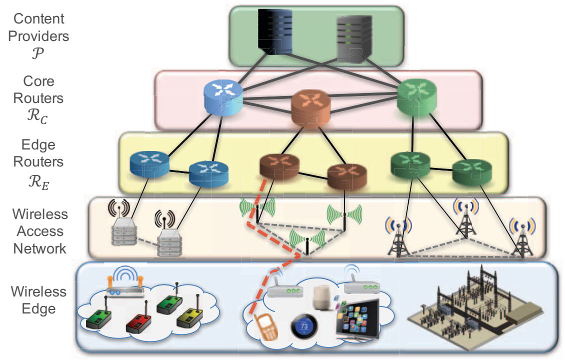

Content Providers $\mathcal{P}$
Core Routers $\mathcal{R}_C$
Edge Routers $\mathcal{R}_E$
Wireless Access Network
Wireless Edge

Fig. 1: Network Architecture: Heterogeneous connected devices at the wireless edge, core and edge routers, and providers.

B. Security Assumptions

We assume that contents are encrypted by the providers. One of the first packets requested by the client contains the key that a client can decrypt using a provider given key. We assume the existence of a public key infrastructure (PKI) by which routers store the providers' public keys and certificates, which will be used for verification procedures. We further assume that symmetric and asymmetric key operations and decryption are secure. We define public key locator as a name that points to a packet that contains the public key or/and its digest.

A legitimate client may share her tag with an unauthorized user. However, we assume the client and the unauthorized user are not co-located under the same access point. It is difficult to differentiate this case from where a client uses multiple devices that are connected to the same access point (e.g. family members sharing an account).

C. Threat Model

In TACTIC, a legitimate client (hereafter client) needs to register its credentials with a content provider and securely obtain an *authentication tag* (hereafter tag). The client then uses her tag for requesting content. Routers authenticate and authorize clients using the tags that are attached to the requests.

The main threat against any access control mechanism is unauthorized access to the content. There are several attack scenarios in this context including: *(a)* a malicious user, requesting a private content without possessing a tag. *(b)* An attacker, requesting a content using a *fake* tag. A tag is fake if it is not generated by a provider or includes invalid fields. *(c)* A client, trying to obtain a content with an expired tags. *(d)* A client, possessing a tag with insufficient access levels. *(e)* A client, sharing his tag with an unauthorized user to allow unauthorized content access. *(f)* An unreliable router that delivers a content to unauthorized users.

In Section 6, we will discuss how TACTIC addresses these threats.

4. Overview of Interactions in TACTIC

In this section, we introduce TACTIC's building blocks along with its brief overview. Table I presents the notations we used to describe our framework.

A. Client-Provider Interaction

In TACTIC, a client $u \in \mathcal{U}$ registers her credential with a content provider $p \in \mathcal{P}$ to obtain an authentication tag that will be added to the subsequent data requests (Interests). When p receives a tag request, it verifies client u's credentials and provides her a fresh tag if she is authorized or drops the request otherwise.

Tag Definition: A tag is a 6-tuple composed of the *provider's public key locator (Pub_p)*, the *client's public key locator (Pub_u)*, the *client's access level (AL_u)*, the *client's access path (AP_u)*, and an *expiry time (T_e)*, and is represented as $\mathcal{T}_{pu} =< Pub_p, AL_u, Pub_u, AP_u, T_e >$. When u's tag expires, she needs to request a fresh tag from the provider. The provider

TABLE I: Notations Used

Notation	Description
$\mathcal{P}$	Set of content providers
$\mathcal{R}_C$	Set of core routers
$\mathcal{R}_C^c$	Set of content routers
$\mathcal{R}_C^i$	Set of intermediate routers
$\mathcal{R}_E$	Set of edge routers
APs	Set of wireless access points
$\mathcal{T}_{pu}$	Client u's tag $\mathcal{T}$ from provider p
Pub_p	Provider p's public key locator
Pub_u	Client u's public key locator
AL_u	Client u's access level
AP_u	Client u's access path
T_e	Tag's expiry time
$T_{current}$	Current time
D	Data packet
$N(..)$	Name prefix extraction function
BF	Bloom filter
FPP	Bloom filter's false positive probability

generates a new tag, signs it to guarantee its integrity and provenance, and sends it to u.

The Pub_p will be used for tag signature verification and will be compared with the provider's public key locator in the content. The AL_u indicates the client u's access level with respect to the content provider. For successful content retrieval, the u's access level in the tag ($AL_u^{\mathcal{T}}$) should satisfy the content access level (AL^D), embedded in the content ($AL^D \leq AL_u^{\mathcal{T}}$). To prevent the impersonation attack and to enforce access control, clients have to sign their requests. Thus, the client's public key locator (Pub_u) is included in the tag, allowing u's corresponding edge router ($r_E \in \mathcal{R}_E$) to use it for u's authentication and authorization.

To avoid the expensive signature verification, we introduce the *access path (AP_u)* feature. AP_u allows the edge router (r_E) to authenticate u's (and her tag's) location, preventing u from sharing her tag with unauthorized users in other locations. Client u's access path (AP_u) is the XOR of the hashed identity of all network entities between u and r_E (excluding r_E). Each intermediate entity, between u and her corresponding r_E, adds its identity to the rolling hash. When provider p receives u's registration request, it adds u's access path (AP_u) to the tag. To authenticate u, r_E compares the AP_u in the request packet with the AP_u in the tag and successfully authenticates u if these hash values are equal. A mobile client needs to request a new tag every time she moves to a new location.

The number of components in a name is defined by the naming schema. However, we believe that the public key name may not have more than two or three components, which makes a tag to be a couple hundred bytes.

B. Procedures Adopted by Routers

As the client's request propagates in the network towards the provider, at least one router ($\mathcal{R}_E \cup \mathcal{R}_C$) on the path validates the tag's integrity and provenance. While a validating router drops a request with an invalid tag, a valid and authentic tag retrieves the requested data (D). In TACTIC, routers use *NACK* packets to express tags' invalidity. A content router ($r_C^c \in \mathcal{R}_C^c$), that receives an invalid request, forwards a NACK to its downstream nodes for preventing the corresponding r_E from delivering the content to the client with the invalid tag. The r_C^c also sends the content along with the NACK to allow the downstream routers to use this content for satisfying their valid pending requests (requests aggregated in the downstream routers' PITs).

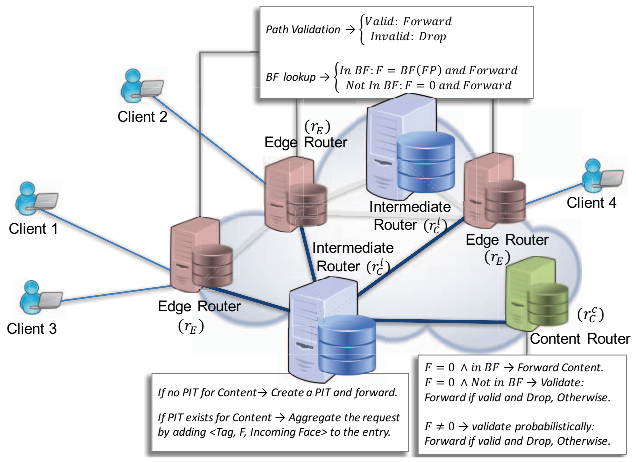

Path Validation → Valid: Forward / Invalid: Drop
BF lookup → In BF: F = BF(FP) and Forward / Not In BF: F = 0 and Forward
Client 2
(r_E) Edge Router
Intermediate Router (r_C^i)
Client 4
Client 1
Intermediate Router (r_C^i)
Edge Router (r_E)
Client 3
Edge Router (r_E)
(r_C^c) Content Router
If no PIT for Content→ Create a PIT and forward.
If PIT exists for Content → Aggregate the request by adding <Tag, F, Incoming Face> to the entry.
F = 0 ∧ in BF → Forward Content.
F = 0 ∧ Not in BF → Validate: Forward if valid and Drop, Otherwise.
F ≠ 0 → validate probabilistically: Forward if valid and Drop, Otherwise.

Fig. 2: Procedures for r_E, r_C^i, and r_C^c.

To eliminate redundant tag validations and reduce the cost of signature verification, in TACTIC, we equip each router with a Bloom filter (BF). A router verifies a received tag's signature and inserts the tag to its BF if the signature is valid. This allows the router to perform cheaper BF lookup operations for the majority of the subsequent requests rather than the more expensive redundant signature verifications.

C. Mechanism Overview

TACTIC is composed of three protocols running on $\mathcal{R}_E$, $\mathcal{R}_C^i$, and $\mathcal{R}_C^c$. Routers in $\mathcal{R}_E$ collaborate with routers in $\mathcal{R}_C$ to eliminate redundant packet verifications and to distribute the verification load across all routers. This cooperation happens by $r_E \in \mathcal{R}_E$ setting a flag (F) in each request they forward. The value of F is set to zero if the received tag is not available in r_E's BF and set to the false positive rate of r_E's BF otherwise (refer to Fig. 2). When a content router receives a request with $F = 0$, it looks for the tag in its Bloom filter and returns the content if it exists. The content router validates the received tag if $F = 0$ but the tag is not available in it Bloom filter. If F is not zero (the edge router has validated the tag), the content router validates the received tag with probability F (false positive of r_E's BF). This ensures that if the r_E's Bloom filter false positive increases, then the probability of a content router validating the tag increases.

As in Fig. 2, each $r_C^i \in \mathcal{R}_C^i$ (bigger nodes shaded in blue) is responsible for aggregating the clients' interests. Following the conventional CCN & NDN model, r_C^i creates a PIT entry when it receives the first request for a packet. Then it aggregates subsequent requests for that content by adding a 3-tuple composed of the request's tag ($\mathcal{T}$), flag F, and the incoming face into the existing PIT entry. When r_C^i receives a content, it forwards the content to the corresponding downstream router(s) and uses these information for validating the aggregated requests in its

PIT. Fig. 2 illustrates an overview of the routers' protocols, which will be discussed in more details in the next section.

5. TACTIC Design

In this section, we present the tag pre-check and routers' protocols in details. First, we explain the tag pre-check protocol, which takes place whenever a router needs to validate a tag. After that, we discuss three protocols: one each for edge routers ($\mathcal{R}_E$), content routers ($\mathcal{R}_C^c$), and intermediate routers ($\mathcal{R}_C^i$), respectively. Protocol 1 presents our low-cost tag pre-check protocol that is employed by routers in $\mathcal{R}_E$ and $\mathcal{R}_C^c$ to validate the received tag using the tag's AL_u, expiry time (T_e), and provider's name prefix before the more expensive BF lookup and signature verification operations.

For successful content retrieval, an r_E compares the provider's name prefix $N(Pub_p^{\mathcal{T}})$, extracted from the tag $\mathcal{T}$'s provider key locator, with the content name prefix, $N(D)$, extracted from the interest (Protocol 1, Lines 1-2). This procedure prevents u from using $\mathcal{T}$ (or more exactly $\mathcal{T}_{pu}$) to retrieve a content from another provider p'. TACTIC leverages tag expiration as the mean to revoke clients' memberships. An r_E, after validating the requested name prefix, examines the tag's expiry (T_e) and drops the request if T_e is less than the current time ($T_{current}$) (Protocol 1, Line 3-4). A shorter expiry time mandates clients to request fresh tags more frequently, which allows a more fine-grained and flexible client revocation.

We set the AL^D (of a publicly available data) to NULL, which allows an r_C^c to return the requested content without tag verification. We envision a hierarchical access level model in which tags with higher access levels can retrieve content with lower access levels ($AL^D \leq AL_u^{\mathcal{T}}$). Therefore, if $AL^D > AL_u^{\mathcal{T}}$ in the received request, the content router drops the request (Lines 8-9). The provider's public key locator in the content packet (Pub_p^D) and tag ($Pub_p^{\mathcal{T}}$) should match for a successful content retrieval (Lines 10-11). The universe of providers that require access control for their clients is significantly small and would potentially number in a few thousands. Thus, our approach of storing public key of the providers would not suffer from scalability issues. We will discuss the benefit of provider's public key matching in Section 6 in more details.

Protocol 1 Pre-check Procedure for Validating Tag $\mathcal{T}$

```
{At Edge Router}
1:  if $N(Pub_p^{\mathcal{T}}) \neq N(D)$ then
2:      $\mathcal{T}$ is invalid;
3:  else if $T_e < T_{current}$ then
4:      $\mathcal{T}$ is invalid;
5:  else
6:      continue;
7:  end if
{At Content Router}
8:  if $AL^D > AL_u^{\mathcal{T}}$ then
9:      $\mathcal{T}$ is invalid;
10: else if $Pub_p^D \neq Pub_p^{\mathcal{T}}$ then
11:     $\mathcal{T}$ is invalid;
12: else
13:     continue;
14: end if
```

Protocol 2 Edge Routers ($r_E \in \mathcal{R}_E$) Procedure

```
{On Interest Arrival}
1:  if $AP_u^{\mathcal{T}} \neq AP_u^{r_E}$ then
2:      Drop the request $\wedge$ send NACK to $u$
3:  else
4:      if $\mathcal{T}_u \in BF^{r_E}$ then
5:          $F = BF^{r_E}(FPP)$
6:      else
7:          $F = 0$;
8:      end if
9:      forward the request
10: end if
{On Content Arrival}
11: if $D == \mathcal{T}_u^{new}$ then
12:     insert $\mathcal{T}_u^{new}$ into $BF^{r_E}$ $\wedge$ forward $D$ to $u$
13: else if $D$ arrives without NACK then
14:     if $F == 0$ then
15:         insert $\mathcal{T}_u$ into $BF^{r_E}$ $\wedge$ forward $D$ towards $u$
16:     else
17:         forward $D$ towards $u$
18:     end if
19: else if $D$ arrives with NACK then
20:     drop the request for $\mathcal{T}_u$
21: end if
22: $\forall \mathcal{T}_w \in PIT(D)$ validate $\mathcal{T}_w$
23: forward $D$ to $w$ if valid and drop otherwise
```

A. Edge Router Protocol

Protocol 2 presents the procedure that an $r_E \in \mathcal{R}_E$ adopts, when it receives an interest and the corresponding response (client u; Provider p).

On Interest: When an r_E receives a request, it first compares the tag's access path ($AP_u^{\mathcal{T}}$) to the one in request ($AP_u^{r_E}$). If $AP_u^{\mathcal{T}} \neq AP_u^{r_E}$, then r_E drops the request and sends a NACK to the client u (Lines 1-2). If $AP_u^{\mathcal{T}}$ is valid, r_E looks up $\mathcal{T}_u$ in its BF (BF^{r_E}) (Line 4). If $\mathcal{T}_u$ exists in BF^{r_E}, r_E sets the value of F to its BF's false positive probability value ($BF^{r_E}(FPP)$) (Line 5). If it is not in BF^{r_E}, r_E sets $F = 0$ (Line 7). Then it forwards the request to its upstream router (Lines 8-9). The value of F will be used by content routers to decide whether to re-validate $\mathcal{T}_u$ or not. We will discuss it in more details in Protocol 3.

On Content: If a registration response packet arrives at r_E for client u ($\mathcal{T}_u^{new}$), i.e. a new tag coming from the producer, r_E adds it into BF^{r_E} and forwards it towards u (Lines 11-12). Otherwise, if the response packet is a content packet without any NACK attached to it, r_E checks the F value that has been set by the content router. If $F == 0$, then $\mathcal{T}_u \notin BF^{r_E}$ at the interest forwarding time; hence, r_E inserts the tag into its Bloom filter and forwards D towards u (Lines 14-15). Any other value of F shows that $\mathcal{T}_u \in BF^{r_E}$, and hence r_E forwards D to u (Line 17). This helps the edge routers to reduce the number of Bloom filter insertion operations, which cumulatively is expensive.

In case that r_E receives a content packet that includes a NACK, it drops the request with $\mathcal{T}_u$ from its PIT (Lines 19-20). Then r_E forwards the D from provider p towards another client w if $\mathcal{T}_w$ exists in its PIT and $\mathcal{T}_w \in BF^{r_E}$, otherwise it validates $\mathcal{T}_w$'s signature before inserting the $\mathcal{T}_w$ in the Bloom filter and forwarding the content towards w (Lines 22-23).

B. Content Router Protocol

On receiving an interest with tag $\mathcal{T}_u$, a content router r_C^c checks the F value in the request (refer Protocol 3). If F is zero, indicating that the corresponding r_E could not validate $\mathcal{T}_u$, r_C^c looks up $\mathcal{T}_u$ in its Bloom filter ($BF^{r_C^c}$). If $\mathcal{T}_u$ exists in the Bloom filter, r_C^c sets the F value of the content D to zero and returns the content-tag pair (Lines 1-3). Otherwise ($\mathcal{T}_u \notin BF^{r_C^c}$), r_C^c validates $\mathcal{T}_u$ (Lines 4-5). Upon successful validation, r_C^c inserts $\mathcal{T}_u$ into its Bloom filter, sets F to zero for reminding r_E that the tag is not available in its Bloom filter, and returns the content-tag pair (Lines 6-9).

In an alternative scenario, $F \neq 0$, r_C^c either decides to re-validate $\mathcal{T}_u$ with probability equivalent to $F = BF^{r_E}(FPP)$ or trust r_E (Line 12). *The rationale behind re-validating an already validated tag is to prevent delivery of the content to an invalid request (tag), which has been forwarded due to a false positive in the edge router's Bloom filter.* For replying with the content, r_C^c copies the received F value from the request to the content packet D and returns the content-tag pair (Lines 13-16). This prevents the corresponding edge router from re-inserting $\mathcal{T}_u$ into its Bloom filter (refer to Protocol 2). If $\mathcal{T}_u$ fails the validation process, r_C^u returns the content-tag-NACK tuple to inform downstream routers on the invalidity of $\mathcal{T}_u$ (Lines 17-19). In TACTIC, r_C^c returns the content D even if $\mathcal{T}_u$ is invalid. This is to satisfy other possible valid aggregated requests in the downstream routers.

C. Intermediate Router Protocol

Protocol 4 presents the procedure for request aggregation and content forwarding at the intermediate routers ($\mathcal{R}_C^i$); routers that have not cached the requested content.

On Interest: On request arrival, an intermediate router (r_C^i) creates a PIT entry and forwards the request if there is no PIT entry for content D (Lines 1-2). However, if the PIT exists, r_C^i adds the tuple $< \mathcal{T}_u, F, InFace_u >$ to the existing PIT entry (Lines 3-5). The existing NDN implementation inserts the entire interest into the PIT entry for aggregation. We note that the addition of the tag adds an overhead to the PIT entry but it is of the order of a couple hundred bytes.

On Content: When r_C^i receives D without a NACK, it returns the content-tag pair towards u over interface $InFace_u$ (Lines 6-7). In case that a NACK is attached to D, r_C^i forwards the content-tag-NACK tuple over $InFace_u$ (Lines 8-10). After handling the arrived content, r_C^i needs to validate the aggregated tags ($\mathcal{T}_w$) for content D in the PIT (Line 11). For each aggregated tuple, if $F \neq 0$ and r_C^i decides not to re-validate $\mathcal{T}_w$, it sends the content on the corresponding face ($InFace_w$) (Lines 12-13). A core router re-validates a tag, when $F \neq 0$, with the false positive probability of the corresponding edge router's Bloom filter that can be obtained from F. However, if $F == 0$ or r_C^i decides to re-verify the tag (with probability corresponding to F), it needs to validate $\mathcal{T}_w$'s signature (Lines 14-15). If $\mathcal{T}_w$ is valid, r_C^i sets F to zero (if it was zero), inserts $\mathcal{T}_w$ into its Bloom filter, and forwards the content-tag pair towards client w on $InFace_w$ (Lines 16-21). If $\mathcal{T}_w$ is invalid, r_C^i forwards the content-tag-NACK tuple on $InFace_w$ towards w (Lines 22-24).

Protocol 3 Content Routers $r_C^c \in \mathcal{R}_C^c$ Procedure

```
1:  if $F == 0 \wedge \mathcal{T}_u \in BF^{r_C^c}$ then
2:      set $F = 0$ in $D$;
3:      return $< D, \mathcal{T}_u >$
4:  else if $F == 0 \wedge \mathcal{T}_u \notin BF^{r_C^c}$ then
5:      validate $\mathcal{T}_u$'s signature
6:      if $\mathcal{T}_u$ is valid then
7:          insert $\mathcal{T}_u$ into $BF^{r_C^c}$
8:          set $F = 0$ in $D$
9:          return $< D, \mathcal{T}_u >$
10:     end if
11: else if $F \neq 0$ then
12:     validate $\mathcal{T}_u$ [with probability $F$]
13:     set received $F$ value in $D$
14:     if $\mathcal{T}_u$ is valid $\vee$ decide not to validate then
15:         return $< D, \mathcal{T}_u >$
16:     end if
17: else
18:     return $< D, \mathcal{T}_u, NACK >$
19: end if
```

D. Case Study

For a better explanation of the request processing procedures, we use Fig. 3, with five clients requesting the same content using their corresponding requests (R1, R2, R3, R4, R5) containing unique tags, T1, T2, T3, T4, T5, respectively. The clients' numbers indicate the order in which clients request the

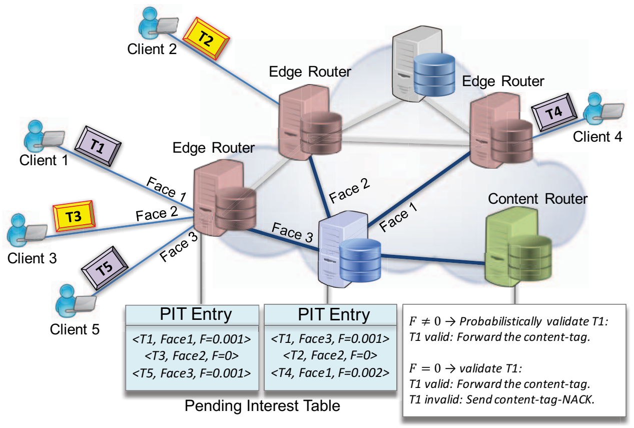


Fig. 3: Request processing routines on the edge (shaded in orange), intermediate (shaded in blue), and content (shaded in green) routers.

Protocol 4 Intermediate Routers $r_C^i \in \mathcal{R}_C^i$ Procedure

```
{On Interest Arrival}
1:  if $\nexists PIT$ entry for $D$ then
2:      create a PIT entry $\wedge$ forward request
3:  else
4:      add $< \mathcal{T}_u, F, InFace_u >$ into the PIT entry;
5:  end if
{On Content Arrival}
6:  if $D$ arrives without NACK then
7:      return $< D, \mathcal{T}_u >$ on $InFace_u$
8:  else if $D$ arrives with NACK then
9:      return $< D, \mathcal{T}_u, NACK >$ on $InFace_u$
10: end if
11: for $\mathcal{T}_w \in PIT$ requesting $D$ do
12:     if $F \neq 0 \wedge$ decide not to re-validate then
13:         return $< D, \mathcal{T}_w >$ on $InFace_w$
14:     else if decide to re-validate $\vee$ $F == 0$ then
15:         validate $\mathcal{T}_w$ [with probability $F$]
16:         if $\mathcal{T}_w$ is valid then
17:             insert $\mathcal{T}_w$ into $BF^{r_C^i}$
18:             if $F == 0$ then
19:                 set $F = 0$ in $D$
20:             end if
21:             return $< D, \mathcal{T}_w >$ on $InFace_w$
22:         else if $\mathcal{T}_w$ is invalid then
23:             return $< D, \mathcal{T}_w, NACK >$ on $InFace_w$
24:         end if
25:     end if
26: end for
```

content. For ease of illustration, we substitute the requests with their corresponding tags in the figure. The purple shaded tags (T1, T4, and T5) are those tags that have been validated by their corresponding edge routers in prior exchanges and have been inserted into their Bloom filters. As a result, the edge routers set the F values of the requests with these tags to non-zero values. Yellow colored tags (T2 and T3) are not available at edge routers' Bloom filters and consequently the requests with these tags will be assigned with zero F values.

The edge router corresponding to Clients 1, 3, and 5 aggregates tags T3 and T5 with the PIT entry for the request with tag T1. The intermediate router that is connected to three edge routers (shaded in blue color) aggregates the requests from Client 2 and Client 4 with the entry that has been generated by Client 1's request. The content router (shaded in green), on receiving Client 1's request, either decides to return the content or validate tag T1 with low probability (F for T1 is equal to edge router's false positive error rate).

Fig. 4 illustrates the content delivery procedures of different routers. On receiving the client's request, the content router either re-validates the tag T1 with a low probability (according to the value defined by the corresponding edge router) or forwards the content without further validation. In either case, the content router forwards the requested content towards its downstream router, the intermediate router that is shaded in blue. On receiving the content, the intermediate router forwards the content on Face 3 towards Client 1 if there is no NACK attached to the content. In case that a NACK is attached to the content, the intermediate router sends the content-tag-NACK tuple towards its downstream router to inform the invalidity of the attached tag. As for Client 4, the intermediate router either probabilistically re-validates the tag or trusts T4 (as it is available in its edge router's Bloom filter) and forwards the content over Face 1.

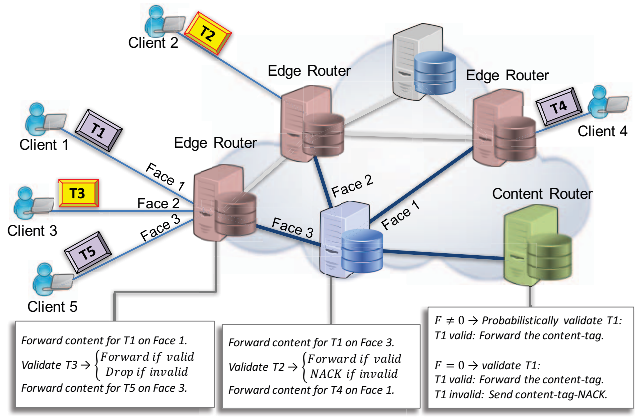

Client 2
T2
Edge Router
Edge Router
T4
Client 4
Client 1
T1
Edge Router
Face 1
T3
Face 2
Client 3
Face 3
Face 2
Face 1
Content Router
Face 3
T5
Client 5
Forward content for T1 on Face 1.
Validate T3 → {Forward if valid / Drop if invalid}
Forward content for T5 on Face 3.
Forward content for T1 on Face 3.
Validate T2 → {Forward if valid / NACK if invalid}
Forward content for T4 on Face 1.
F ≠ 0 → Probabilistically validate T1:
T1 valid: Forward the content-tag.
F = 0 → validate T1:
T1 valid: Forward the content-tag.
T1 invalid: Send content-tag-NACK.

Fig. 4: Data processing routines on the edge (shaded in orange), intermediate (shaded in blue), and content (shaded in green) routers.

However, the intermediate router has to validate Client 2's tag (T2) since it has not been validated before. In case that the tag is valid, the intermediate router forwards the content over Face 2 towards Client 2; the corresponding edge router inserts Client 2's tag (T2) into its Bloom filter. Otherwise, it sends the content and a NACK to Client 2's edge router. The edge router connected to the three clients, forwards the content towards Client 1 and Client 5 on Face 1 and Face 3, respectively. As for Client 3, her corresponding edge router (connected to Client 3) validates tag T3 and forwards the content upon successful validation followed by adding T3 to its Bloom filter. If the tag is invalid, the edge router drops the PIT entry for Client 3.

6. TACTIC Security and Special Cases

TACTIC's security is achieved by combining the tag validation and path authentication procedures. On one hand, the Bloom filter assisted tag validation verifies the tags integrity, provenance, and freshness. On the other hand, the access path based authentication prevents unauthorized users in other locations from accessing the content.

Before discussing TACTIC's security in more details, we briefly explain how the clients can decrypt the encrypted content. A provider can encrypt the content decryption key with the client's public key and send it to the client along with her tag. A revoked user, despite possessing a tag and its decryption key, cannot retrieve the content from the network as the tag is expired. A malicious ISP router can collude with a revoked client to deliver him the encrypted content, which breaks TACTIC's security. The revoked client can use the content decryption key from expired tags for decrypting the content. But, compromising ISP routers is difficult.

A. Malicious Client

As we mentioned in Subsection 3.3, there are threats against our mechanism, such as content retrieval using expired tags, fake tags, or unauthorized use of shared or replayed tags. In TACTIC, any router can validate a tag by verifying the provider's signature thus preventing unauthorized access to content. In our design, using Bloom filters for storing validated tags allows the routers to reduce the cost of signature verification to a *constant time* Bloom filter look up. To further reduce the routers computational overhead, edge and content routers perform pre-filtering procedures for detecting the invalid tags before Bloom filter look up.

In TACTIC, edge routers drop a request if the name prefix of the requested content does not match the tag's name prefix field to prevent a client using a valid tag of Provider "A" to retrieve a content from Provider "B." The edge routers drop the requests with expired tags to prevent the propagation of expired requests into the network core. Also, an edge router validates the access path from the request with the tag's access path to guarantee that an unauthorized user cannot access the content, unless it is co-located with an authorized client under the same access point. Note that it is difficult to differentiate a client using multiple instances of the same application on her device from co-located attacker(s) and client.

In our mechanism, the content routers (routers caching content) are responsible for checking whether the tag's access level can satisfy the requested content's access level. Other than the access level, the content routers match the provider's public key in the tag and the content for validation.

B. Malicious Content Provider

A malicious content provider might hijack a legitimate provider's name prefix to poison the network with fake content or prevent clients service access. To do so, the malicious provider generates a malicious tag for a client, which will be used for content retrieval. We define a malicious tag as one that either includes a legitimate provider's public key locator or the malicious provider's public key locator, and is signed by the malicious provider. The content router detects the malicious tag if the malicious provider's public key is included in the tag. In case that the legitimate provider's public key is included in the tag, the malicious tag fails the signature verification. However, a false positive at an edge router's Bloom filter allows the request with malicious tag to be forwarded in the network. The intermediate router, if they do not aggregate the interest, will not validate the tag. The tag also may not be validated by a content router. Then the fake content will be delivered to the client. We note that this scenario will be extremely rare, and even if this happens the client can validate the content and drop it. In this scenario, the client can validate the content by verifying its signature.

One can see that a malicious provider is not a threat to TACTIC's security. However, a legitimate client's attempts for retrieving a content with a malicious tag will be unsuccessful, which causes service degradation on the client. Note that a client might receive a malicious tag only if the routers' FIBs are populated incorrectly (pointing to a malicious provider). With secure routing protocols, this is rare, but this can affect all communications.

Note that a malicious content provider can orchestrate a network DoS attack by adjusting its tags validity to a short period (e.g., one second). In such a scenario, the clients have to request fresh tags every second. However, obtaining a fresh tag only requires one request per client, which is negligible compared to the large number of requests for actual content retrieval–essentially a low-rate DoS attack.

7. Comparison with the State-of-the-Art

In this section, we compare TACTIC with the state-of-the-art in ICN's access control in terms of fundamental features including client revocation, communication/computation cost, and dependency on additional infrastructure. Table II presents the summary of this comparison.

Almost all ICN-based access control mechanisms incur communication overhead. The communication overhead can be in forms of extensive communication between network entities [9] or metadata that is needed for content consumption [16]. While a few mechanisms incur constant communication overhead [8], [14], in other approaches the overhead increases with the number of clients [3], [7] or their attributes [10]. Similar to [16], TACTIC's communication overhead is the fixed size tags in the requests, which places it among mechanisms with low communication overhead. Adding tags to requests also eliminates the needs of an always-online authentication server similar to [3], [7].

Access control delegation to the network introduces additional computation burden to entities such routers and proxies [8], [10]. In TACTIC, similar to [8], the network entities perform Bloom filter operations in addition to infrequent signature verifications and access path validations. However, these operations are less compute intensive compared to costly cryptographic computation proposed in [3], [5], [7], [10], [11]. Furthermore, unlike [8], TACTIC can satisfy the authorized requests for the cached content without relying on a proxy for access policy decryption [10].

One of the most important features of any access control mechanism is efficient and effective client revocation. In this context, efficiency measures the additional communication and computation cost of revoking a client, while effectiveness measures how fast the system can eliminate the revoked access to the content. Some mechanisms [5], [10], [11] require content re-encryption for each revocation–a prohibitive practice. Others [3], [7] require meta-data generation and network-wide distribution. In contrast to these mechanisms, the cost of client revocation in TACTIC is reduced to a tag request/response communication for each client, which is significantly lower than the above approaches. TACTIC's flexible revocation mechanism allows the content provider to set a shorter tag expiry time,

TABLE II: TACTIC's Comparison with the Proposed Access Control Mechanisms

Mechanism	Communication Overhead	Computation Burden			Additional Infrastructure	Client Revocation	Access Control Enforcement
		Provider	Network	Client			
TACTIC	Low	-	Low	-	N/A	Tunable Time-based	Network
Misra *et al.* [3], [7]	Moderate	-	-	Moderate	N/A	Threshold Based	Client
Chen *et al.* [8]	Low	High	Low	-	N/A	Daily Re-encryption	Provider
Kurihara *et al.* [9]	High	High	Moderate	-	Required	Lazy Revocation	Provider
Da Silva *et al.* [10]	Low	-	High	-	Required	Key Update per Revoc.	Network
Hamdane *et al.* [11]	Low	High	-	Moderate	N/A	System Re-key	Provider
Li *et al.* [4], [12]	Moderate	Moderate	-	Moderate	Required	N/A	Client
Wood *et al.* [14]	Low	High	-	-	N/A	N/A	Provider
Mangili *et al.* [5]	Low	High	-	Moderate	N/A	Partial Re-encryption	Client
Tan *et al.* [15]	High	Extreme	-	-	N/A	Provider Authentication	Provider
Li *et al.* [16]	Low	Moderate	Low	-	N/A	N/A	Provider

resulting in more frequent tag expiry, which is more effective compared to the daily revocation proposed in [8].

8. Simulation Results and Analysis

In this section, we first explain our simulation setup and the scope of implementation and then present the evaluation criteria, our simulation results, and numerical analysis.

A. Simulation Setup

We extended the ndnSIM-2.3 simulator [17] (an ns-3 module) to implement TACTIC. We implemented the Client Registration Procedure in which a client uses her credentials to obtain a tag from the producer. We further implemented the Tag Pre-check Procedure (*Protocol 1*), the Edge Router Procedure (*Protocol 2*), the Content Router Procedure (*Protocol 3*), and the Intermediate Router Procedure (*Protocol 4*).

Network Setup: We ran our simulations on four different scale free network topologies with 500Mbps (1 ms latency) core and 10Mbps (2 ms latency) edge links. We selected a few designated routers from these topologies as the edge routers to add clients, attackers, and providers. Table III presents the number of routers, providers, legitimate clients, and unauthorized users (hereafter attackers) in our topologies. We randomly selected the number of attackers to be roughly one-third and the legitimate clients to be the two-third of the user base.

In our implementation, each router is equipped with a BF. The FPP of a BF has been formalized given its size, the number of hash functions, and the number of items to be indexed [18]. For simulation purposes, we set the BF to index $500, 1000, 1500$ tags, the number of hash functions as 5, and the maximum FPP as 0.0001. To avoid additional false positives, in our implementation, each router automatically resets its BF, when its is saturated (its FPP reaches the maximum FPP).

Client and Attacker Setup: We implemented a *Zipf-window client* in which each client is equipped with a fixed size window for outstanding requests (set to 5 requests in our simulations).

TABLE III: Network Topologies with the Number of entities.

	Topo. 1	Topo. 2	Topo. 3	Topo. 4
Core Routers	80	180	370	560
Edge Routers	20	20	30	40
Providers	10	10	10	10
Legitimate Clients	35	71	143	213
Attackers	15	29	57	87

Clients take the content popularity (Zipf distribution with $\alpha = 0.7$) into account to select and request new contents. Clients first register themselves at the content providers, if they do not possess any valid tag from that providers, and then request the selected contents.

We implemented the attackers that we defined in the threat model (Subsection 3.3). However, we left the implementation of the access path feature as part of our future work. Attackers are also equipped with outstanding request windows.

Content Producer Setup: We set each producer to generate 50 content objects of 50 chunks each. Content popularity follows a static Zipf distribution ($\alpha = 0.7$), popularity does not change over time [19]. The expiry time of a tag was set to 10 seconds to investigate scenarios with a high tag churn rate, which creates computation overhead at the routers.

Evaluation Criteria: We use *user-based* and *network-based* metrics to evaluate TACTIC. The user-based metrics are: *(i) average content retrieval latency*, *(ii) request satisfaction ratio*, and *(iii) tag statistics* (number of requested/received tags).

The network-based metrics are: *(i) computational overhead* such as BF insertions, look ups, and signature verifications and *(ii) BF reset threshold*. We define the BF reset threshold as the minimum number of requests that cause a BF to reach its maximum FPP threshold, which prompts the BF's reset.

B. Results and Analysis

We ran our simulations for 2000 seconds and averaged the results of each topology over five runs with different seeds. The ns-3 (and hence ndnSIM) simulator does not take the time of the computational operations into account. Thus, we benchmarked the latency distribution (normal distribution) of our computation-based events: BF look up $\sim \mathcal{N}(9.14\times10^{-7}, 6.51\times10^{-9})$, BF insertion $\sim \mathcal{N}(3.35 \times 10^{-7}, 1.73 \times 10^{-3})$, and signature verification $\sim \mathcal{N}(1.12 \times 10^{-5}, 6.49 \times 10^{-3})$, on a machine (with Intel Core-i7, 2.93GHz, 20.5GB RAM) running Ubuntu 14.04. This allowed us to apply the delays, for computation-based operations, as random variables according to our benchmarks.

Fig. 5 illustrates the content retrieval latency (averaged per second) of our mechanism with three BF sizes. The average content retrieval latency decreases as the size of the BF increases. This is because the FPP grows slowly with the size

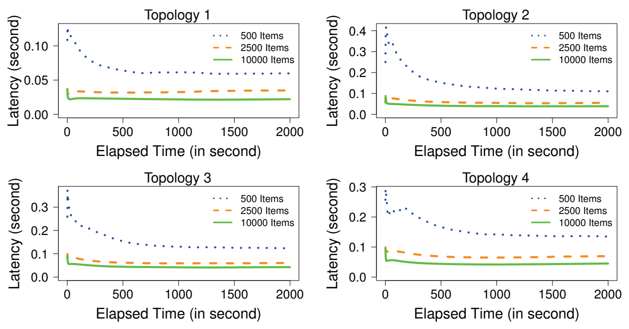


Fig. 5: The clients content retrieval latency for various BF sizes: 500, 2500, and 10000 elements (per-second average).

of BF, causing infrequent BF resets. After each BF reset, the corresponding edge router needs to validate tags and insert them into its BF. This tag re-validation requires a signature verification followed by a BF insertion that contribute to the higher communication latency. Hence, a bigger BF results in lower latency.

To show TACTIC's effectiveness in deterring unauthorized access, we evaluated the successful content delivery ratio of the clients and attackers. As it is shown in Table IV, attackers experienced at most 0.78% successful delivery ratio compare to clients' 99.99% successful delivery ratio. The success rate of clients is not 100% on account of a minimal amount of network packet losses. In our three topologies, Topo 2, Topo 3, and Topo 4, the attackers receive between 10 to 44 content chunks. Our investigation showed that only attackers with invalid signatures were successful in retrieving content, which is caused by BFs' false positives. The rationale behind attackers requesting fewer number of chunks is that their windows are occupied with outstanding requests, which have been dropped in the network due to their invalid tags. The attackers' window size reduces upon request expiry (set to one second), that's when the attackers can request new chunks. This has a secondary advantage of request-based DoS prevention.

TACTIC's computational efficiency has a direct relation with the tags validity period; longer expiry periods result in fewer signature verifications and BF insertions, which reduce the routers computational overhead and clients latencies. However, longer validity periods prevent immediate client revocations, thus increasing the chance of unauthorized content access.

Fig. 6 presents the per-second clients tag-request (Q) and tag-receive (R) rates. These rates increase linearly with the size of topology (and hence the number of clients). The high rate tag-request and tag-receive that we observe are caused by a short tag expiry time (10 seconds), which has been set to

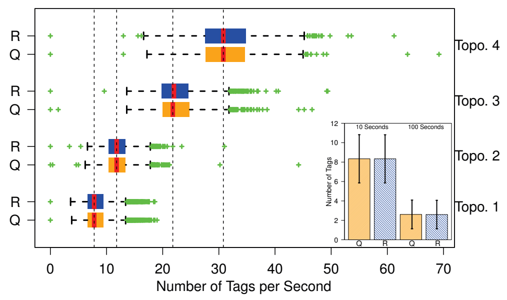


Fig. 6: The tag-request (Q) and tag-receive (R) rates for all clients (averaged per second).

achieve more frequent revocation. We use the inner bar-plot in Fig. 6 to show the effect of a longer tag expiry time on Topology 1. On average, these rates can be reduced to one-fourth by increasing the validity period from 10 to 100 seconds.

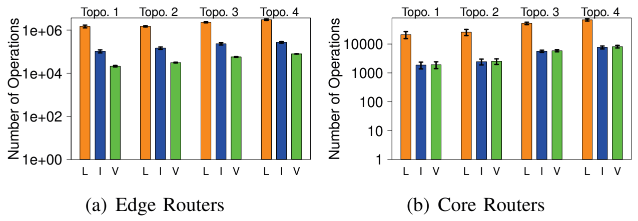


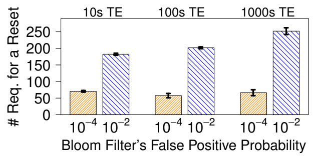
Fig. 7: Number of BF look ups (L), insertions (I), and signature verifications (V) at: (a) edge routers, and (b) core routers.

In TACTIC, providers delegate the authentication and authorization tasks to the routers. Fig. 7 quantifies the number of BF look ups (L), insertions (I), and signature verifications (V) on the edge and core routers. At the edge routers (refer Fig. 7(a)), the BF look up (the cheapest operation) occurs the most while signature verification (the most expensive operation) happens the least (two orders of magnitude less). An edge router inserts a tag in its BF if the tag *(i)* arrives from the client and is valid or *(ii)* arrives from an upstream router that vouches for its validity. In spirit, the number of insertions and signature verifications should be same; tag will be inserted after it is validated. However, it is not the case for the edge routers as the edge routers can also insert the tags that have been validated by upstream routers.

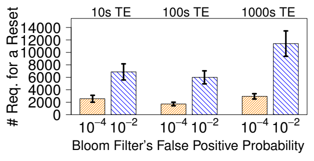
From Fig. 7, we can observe a drastic decrement in the computational overhead of core routers compared to edge routers. The reasons include request aggregation that reduces the core network traffic and the routers collaboration, which reduces redundant tag verification (refer to Protocol 2). Among

TABLE IV: Clients and Attackers Successful Delivery Ratio across Different Topologies.

	Topology 1		Topology 2		Topology 3		Topology 4	
	Client	Attacker	Client	Attacker	Client	Attacker	Client	Attacker
Requested Chunk	1468199	1643	1452635	2207	2287708	4041	3017804	5535
Received Chunk	1468070	0	1452407	10	2287263	10	3017137	44
Delivery Rate	0.9999	0.0	0.9998	0.0044	0.9998	0.0025	0.9997	0.0078



(a) Edge Routers



(b) Core Routers

Fig. 8: # of requests for BF reset with various the false positive probabilities (FPP) and tag expiry (TE) periods.

the three aforementioned operations, BF look up is a function of the client request rate. However, BF insertion and signature verification (more computation intensive operations) occur following edge routers' BF resets. This allows us to reduce the routers workload by reducing the number of BF resets, which is a function of BF size, its FPP, and the tag validity period.

Fig. 8 presents the effect of various FPPs and tag validity periods on the number of requests received by the routers before a BF reset is needed (Topology 1). It is obvious that a higher value is more desired. For a fixed FPP in Fig. 8(a), we can observe that the amount of requests for one BF reset does not considerably change with different tag validity periods. However, increasing the FPP from 0.0001 to 0.01 significantly changes the expected number of requests for a BF reset. As it is shown in Fig. 8(b), the core routers follow the same trend. Even a modest increase in number of requests needed before a reset implies that the routers have to spend significantly less time in signature verifications, thus helping with core scalability.

To explore BF improvement, we increased the size of BFs from 500 to 5000 for two different false positive probabilities. Table V shows the results. We approximately reduced 93% of the edge and 99% of the core routers BF resets by increasing the Bloom filters size to 5000. This results shows the impact of the Bloom filter size compared to its FPP on reducing the routers computational overhead.

TABLE V: Number of BF Resets for Various Size and FPP values with 10 Seconds Tag Expiry Period.

BF Size	500 Items		5000 Items		**Improvement**	
BF FPP	10^{-4}	10^{-2}	10^{-4}	10^{-2}	10^{-4}	10^{-2}
Edge Routers	20840	9354	1233	609	94.08%	93.48%
Core Routers	596	255	8	1	98.65%	99.60%

9. Conclusions

In this paper, we proposed, TACTIC, an access control mechanism for ICN wireless edge, in which the authentication and authorization tasks are delegated to the network entities such as routers and access points. By leveraging Bloom filters, TACTIC dramatically reduces the costly signature verification at the intermediate entities. The client side complexity of TACTIC is only obtaining a fresh tag from the providers upon tag expiry. Our simulation results demonstrates the practicality of our mechanism.

In future, we plan to augment our mechanism with a traitor tracing feature for preventing the clients from sharing their tags with unauthorized users and thwarting replay attack. We also plan to test our mechanism in a real testbed under nodes mobility.

References

[1] "The Zettabyte Era: Trends and analysis," 2017, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html.

[2] "Cisco Visual Networking Index: Global mobile data traffic forecast update, 2016 to 2021 white paper," 2017, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[3] S. Misra, R. Tourani, and N. Majd, "Secure content delivery in information-centric networks: design, implementation, and analyses," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, 2013, pp. 73–78.

[4] B. Li, A. Verleker, D. Huang, Z. Wang, and Y. Zhu, "Attribute-based access control for ICN naming scheme," in *Proceedings of IEEE Conference on Communications and Network Security*, 2014, pp. 391–399.

[5] M. Mangili, F. Martignon, and S. Paraboschi, "A cache-aware mechanism to enforce confidentiality, trackability and access policy evolution in content-centric networks," *Computer Networks*, vol. 76, pp. 126–145, 2015.

[6] R. Tourani, S. Misra, T. Mick, and G. Panwar, "Security, privacy, and access control in information-centric networking: A survey," *IEEE Communications Surveys & Tutorials*, 2017.

[7] S. Misra, R. Tourani, F. Natividad, T. Mick, N. Majd, and H. Huang, "AccConF: An access control framework for leveraging in-network cached data in ICNs," *IEEE Transactions on Dependable and Secure Computing*, 2017.

[8] T. Chen, K. Lei, and K. Xu, "An encryption and probability based access control model for named data networking," in *Proceedings of IEEE International Performance Computing and Communications Conference*, 2014, pp. 1–8.

[9] J. Kurihara, E. Uzun, and C. Wood, "An encryption-based access control framework for content-centric networking," in *Proceedings of IFIP Networking Conference*, 2015, pp. 1–9.

[10] R. S. D. Silva and S. Zorzo, "An access control mechanism to ensure privacy in named data networking using attribute-based encryption with immediate revocation of privileges," in *Proceedings of IEEE Consumer Communications and Networking Conference*, 2015, pp. 128–133.

[11] B. Hamdane, A. Serhrouchni, and S. Fatmi, "Access control enforcement in named data networking," in *Proceedings of International Conference for Internet Technology and Secured Transactions*, 2013, pp. 576–581.

[12] B. Li, Z. Wang, D. Huang, and Y. Zhu, "Toward privacy-preserving content access control for information centric networking," DTIC Document, Tech. Rep., 2014.

[13] M. Raykova, H. Kazmi, H. Lakhani, and A. Gehani, "Decentralized authorization and privacy-enhanced routing for information-centric networks," in *Proceedings of the 31st ACM Annual Computer Security Applications Conference*, 2015, pp. 31–40.

[14] C. Wood and E. Uzun, "Flexible end-to-end content security in CCN," in *Proceedings of IEEE Consumer Communications and Networking Conference*, 2014, pp. 858–865.

[15] X. Tan, Z. Zhou, C. Zou, Y. Niu, and X. Chen, "Copyright protection in named data networking," in *Proceedings of IEEE International Conference on Wireless Communications and Signal Processing*, 2014, pp. 1–6.

[16] Q. Li, X. Zhang, Q. Zheng, R. Sandhu, and X. Fu, "Live: Lightweight integrity verification and content access control for named data networking," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 308–320, 2015.

[17] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2: An updated NDN simulator for NS-3," NDN, Technical Report NDN-0028, Revision 2, 2016.

[18] J. K. Mullin, "A second look at bloom filters," *Communications of the ACM*, vol. 26, no. 8, pp. 570–571, 1983.

[19] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proceedings of the IEEE Computer and Communications Societies*, vol. 1, 1999, pp. 126–134.