KRB-CCN: Lightweight Authentication & Access Control for Private Content-Centric Networks

Ivan O. Nunes and Gene Tsudik

University of California Irvine, USA
{ivanoliv,g.tsudik}@uci.edu

Abstract. Content-Centric Networking (CCN) is an internetworking paradigm that offers an alternative to today's IP-based Internet Architecture. Instead of focusing on hosts and their locations, CCN emphasizes addressable named content. By decoupling content from its location, CCN allows opportunistic in-network content caching, thus enabling better network utilization, at least for scalable content distribution. However, in order to be considered seriously, CCN must support basic security services, including content authenticity, integrity, confidentiality, authorization and access control. Current approaches rely on content producers to perform authorization and access control, which is typically attained via public key encryption. This general approach has several disadvantages. First, consumer privacy vis-a-vis producers is not preserved. Second, identity management and access control impose high computational overhead on producers. Also, unnecessary repeated authentication and access control decisions must be made for each content request. (This burden is particularly relevant for resource-limited producers, e.g., anemic IoT devices.)
These issues motivate our design of KRB-CCN – a complete authorization and access control system for private CCN networks. Inspired by Kerberos in IP-based networks, KRB-CCN involves distinct authentication and authorization authorities. By doing so, KRB-CCN obviates the need for producers to make consumer authentication and access control decisions. KRB-CCN preserves consumer privacy since producers are unaware of consumer identities. Producers are also not required to keep any hard state and only need to perform two symmetric key operations to guarantee that sensitive content is confidentially delivered only to authenticated and authorized consumers. Furthermore, KRB-CCN works transparently on the consumer side. Most importantly, unlike prior designs, KRB-CCN leaves the network (i.e., CCN routers) out of any authorization, access control or confidentiality issues. We describe KRB-CCN design and implementation, analyze its security, and report on its performance.

1 Introduction

Content-Centric Networking (CCN) is an emerging internetworking paradigm that emphasizes transfer of named data (aka content) instead of host-to-host communication [1, 2]. All CCN content is uniquely named. Content *producers*

are entities that publish content under namespaces. Entities that wish to obtain content, called *consumers*, do so by issuing an *interest* message specifying desired content by its unique name. The network is responsible for forwarding the interest, based on the content name, to the nearest copy of requested content. Interests do not carry source or destination addresses. Each interest leaves state in every router it traverses. This state is later used to forward, along the reverse path, requested *content* back to the consumers. As content is forwarded to the consumer, each router can choose to cache it. If a popular content is cached, subsequent interests for it can be satisfied by the caching router and not forwarded further. This can lead to lower delays, better throughput and improved network utilization.

Due to CCN's unique characteristics, security focus shifts from securing host-to-host tunnels to securing the content itself. CCN mandates that each content must be signed by its producer. This is the extent of CCN network-layer security. In particular, CCN does not make any provisions for confidentiality, authorization or access control, leaving these issues to individual applications. We believe that this approach makes sense, since involving the network (i.e., routers) in such issues is generally problematic for both performance and security reasons.

Access control (AC) in CCN has been explored in recent years. Most approaches [3–9] rely on using public key encryption to safeguard content (we overview these approaches in Section 6). Specifically, producers are expected to encrypt content with a public key corresponding to an authorized consumer, or a group thereof. The latter use their corresponding private keys to decrypt. Although it seems to work, this approach exhibits several problems:

- First, producers are responsible for handling consumer authentication and content AC on their own. Thus, they must deal with (1) consumer identity management and authentication, (2) AC policy representation and storage, (3) updates of access rights, and (4) content encryption. In some cases, producers might not want (or be able) to deal with this burden, e.g., resource-constrained IoT devices. On the consumer side, this means keeping track of producer-specific authentication contexts and keys.
- Second, AC enforced by producers implies sacrificing consumer privacy, which is an important and appealing CCN feature. Since CCN interests do not carry source addresses, a content producer (or a router) normally does not learn the identity of the consumer. However, if the producer enforces AC, it learns consumers identities.
- Third, if a set of producers belong to the same administrative domain and each producer enforces its AC policy, it is difficult to react to policy changes, e.g., access revocation for a given consumer or a consumer's credential. Implementing such changes requires notifying each producer individually.
- Finally, since public keys bind authorization rules to identities, authentication of consumers is attained via consumer-owned private keys. However, if consumer is authenticated by other means, e.g., passwords and biometrics, each producer would have to store and manage potentially sensitive state information (password files or biometric templates) for each consumer.

Since mid-1980s, Kerberos [10] has been successfully and widely used to address these exact issues in private IP-based networks or so-called stub Autonomous Systems. Kerberos de-couples authentication and authorization services via short-term *tickets*. It also allows services (e.g., storage, compute or web servers) to be accessed by clients over a secure ephemeral session. By checking a client's ticket for freshness of authentication information, a service limits the period of vulnerability due to revocation.

In this paper, we present KRB-CCN, a system inspired by Kerberos for authentication and access control (AC) enforcement in CCN, that aims at addressing the aforementioned issues. KRB-CCN treats consumer authentication and authorization as separate services. It uses tickets to allow consumers to convey authorization permissions to servers, e.g., content producers or repositories. Servers use tickets to determine whether requested content should be provided. KRB-CCN also introduces a novel namespace based AC policy, which allows a consumer to securely retrieve content without revealing its identity to the content producer, thus preserving consumer privacy. In addition, KRB-CCN is transparent to the users; they need not be aware of KRB-CCN or perform any additional tasks. It is also completely invisible to the network, i.e., CCN routers are unaware of KRB-CCN.

Organization: Section 2 overviews CCN and Section 3 overviews Kerberos. Next, Section 4 introduces KRB-CCN, including its system architecture, namespace based AC scheme, and the protocol for authentication, authorization, and secure content retrieval. In addition, a security analysis for the design presented in Section 4 is provided in Appendix A. Then, performance of KRB-CCN is evaluated in Section 5. Finally, Section 6 discusses related work and Section 7 concludes this paper.

2 CCN Overview

We now overview key features of CCN. Given basic familiarity with CCN, this section can be skipped with no loss of continuity.

In contrast to today's IP-based Internet architecture which focuses on endpoints of communication (i.e., interfaces/hosts and their addresses) CCN [1, 11] centers on content by making it named, addressable, and routable within the network. Moreover, a content must be signed by its producer. A content name is a URI-like string composed of one or more variable-length name segments, separated by the '/' character. To obtain content, a user (consumer) issues an explicit request message, called an *interest* containing the name of desired content. This interest can 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. Name matching is exact, e.g., an interest for /edu/uni-X/ics/cs/fileA can only be satisfied by a content object named /edu/uni-X/ics/cs/fileA.

In addition to a payload, a content object includes several other fields. In this paper, we are only interested in the following three: Name, Validation, and

`ExpiryTime`. `Validation` is a composite of validation algorithm information (e.g., the signature algorithm, its parameters, and the name of the public verification key), and validation payload, i.e., the content signature. We use the term "signature" to refer to the entire `Validation` field. `ExpiryTime` is an optional, producer-recommended duration for caching a content object. Interest messages carry a name, optional payload, and other fields that restrict the content object response. We refer to [11] for a complete description of all CCN message types, fields and their semantics.

Packets are moved within the network by routers. Each CCN router has two mandatory (and one optional) components:

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

A router uses its FIB to forward interests toward the producer of requested content. Whereas, a router uses its PIT to forward content along the reverse path towards consumers. Specifically, upon receiving an interest, a router R first checks its cache (if present) to see if it can satisfy this interest locally. In case of a cache miss, R checks its PIT for an outstanding version of the same interest. If there is a PIT match, the new interest's incoming interface is added to the PIT entry. Otherwise, R creates a new PIT entry and forwards the interest to the next hop according to its FIB (if possible). For each forwarded interest, R stores some state information in the PIT entry, including the name in the interest and the interface from which it arrived, such that content may be returned to the consumer. When content is returned, R forwards it to all interfaces listed in the matching PIT entry and then removes the entry. A content that does not match any PIT entry is discarded.

3 Kerberos Overview

We now summarize Kerberos. We refer to [12] for a more extensive description. Kerberos includes four types of entities: clients, services, an Authentication Server (AS), and a Ticket-Granting Server (TGS). The AS/TGS pair (which are often collocated within the same host) is also known as a Key Distribution Center (KDC). Should a new client/user or a new service be added to the network, it must first be properly registered into KDC's (AS and TGS) databases.

In Kerberos' terminology, a ***realm*** corresponds to a single administrative domain, e.g., a private network or a stub Autonomous System. Each realm has one KDC and any authorization or AC decision by a KDC is only valid within its realm. Thus, identities, tickets, and encryption keys (see below) are also realm-specific.

Principal is the term used to refer to names of entries in the KDC database. Each user/client and service has an associated principal. User principals are generally their usernames in the system. Service principals are used to specify various applications. A service principal has the format: `service/hostname@realm`. A service specification is needed in addition to a hostname, since a single host often runs multiple services. With Kerberos operation in IP Networks, principals are resolved to host IP addresses via DNS look-ups [13]. As can be expected, CCN obviates the need for DNS look-ups, since all content objects are uniquely named by design. Moreover, routing is done based on content names. As discussed below, KRB-CCN enforces AC based on content namespaces, instead of service principals.

Each client/user principal (i.e., username) stored in the AS database is associated with a key, which can be either a public-key or a symmetric key derived from the user's password. Also, the same client/user principal also exists in the TGS database. However, it is associated with a list principals for services that such user has permission to access.

Before attempting to access any content, a client must first authenticate to its local AS. This is done by either typing a password, or proving possession of a secret key associated with the client's identity in the AS database. If the client proves its identity, AS issues a ***Ticket-Granting Ticket (TGT)*** – a temporary proof of identity required for the authorization. This TGT might be cached and used multiple times until its expires.

The client uses a valid TGT to request, from TGS, authorization for a service. The TGS is responsible for access control decisions – verifying whether the requested service is within the set of permitted services for the identity ascertained in the provided TGT. If the result is positive, TGS issues a ***Service Ticket (ST)*** to be used for requesting the actual service.

4 KRB-CCN Design

There are three fundamental requirements for any authentication and authorization system. First, AC policies must effectively bind identities to their access rights. Second, once AC policies are established, there must be a way to enforce them, thus preventing unauthorized access. Third, authentication mechanisms must ensure that identities can not be spoofed; this includes both producers and consumers. The system must also not involve the network elements (i.e., routers) where authentication and authorization burden is both misplaced and simply unnecessary.

In the rest of this section, we describe how KRB-CCN achieves each of these requirements. We start by introducing KRB-CCN system architecture and its namespace-based AC policy, which takes advantage of CCN hierarchical content name structure to provide AC based on content prefixes. Next, we describe KRB-CCN communication protocol, which enforces AC policies while providing a single sign-on mechanism for user authentication. Though KRB-CCN is inspired by Kerberos for IP-based networks, it also takes advantage of unique CCN fea-

tures to effectively satisfy basic authentication and authorization requirements. Throughout the protocol description we discuss the intuition behind the security of KRB-CCN. A detailed security analysis of KRB-CCN is provided in Appendix A.

As it is the case for IP-based Kerberos, KRB-CCN targets private (content-centric) networks, such as intra-corporation/intra-Autonomous Systems settings.

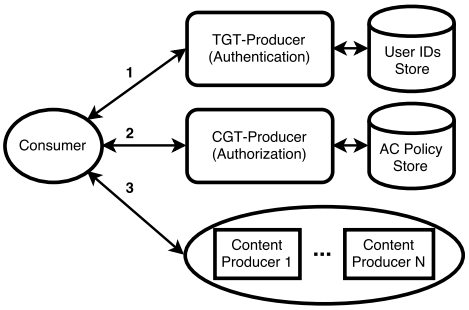


Fig. 1. KRB-CCN system architecture

4.1 System Architecture

Recall that Kerberos has 4 types of entities: AS, TGS, client, and server. A Kerberos realm (domain) typically has one AS/TGS pair, usually collocated in the same host, as well as multiple clients and servers. KRB-CCN also includes four types of entities that map to Kerberos entities as follows:

- Consumer: corresponds to Kerberos client. It issues interests for content and services according to KRB-CCN protocol. Each consumer has an identity and a set of associated permissions registered in the system.
- Producer: subsumes one or more Kerberos servers. A producer is required to register its namespace(s) in the system, by registering with a TGS (see below). A single namespace, i.e., a name prefix, can correspond to a single Kerberos server. Alternatively, a group of namespaces of the same producer can be treated as a single server. A producer does not perform any direct consumer authentication or authorization. A producer only checks whether a content-requesting consumer possesses a valid TGS-issued ticket.
- As in Kerberos, authentication and authorization are handled by two logically separated entities which can be collocated:
  Authentication Server (AS) (aka TGT-Prod): treated as a special type of producer that generates so-called Ticket-Granting Tickets (TGT-s) for consumers once their identities are verified. These tickets can then be used as temporary proofs of identity. We refer to the AS as *TGT-Prod*.

Ticket-Granting Server (TGS) (aka CGT-Prod): performs authorization and is also treated as a special type of producer. Based on a valid consumer TGT and a request to access to a given namespace (server), TGS checks whether this consumer is allowed access to the requested namespace. If so, TGS issues a Content-Granting Ticket (CGT), which proves to the producer that this consumer is granted access to any content under the requested namespace. We refer to TGS as *CGT-Prod*.

Figure 1 illustrates KRB-CCN system architecture. As part of the log-in procedure (aka single sign-on or SSO), a consumer authenticates to TGT-Prod (round 1) and obtains a TGT, which it caches. Whenever a consumer wants to initially access content from a particular producer, it requests authorization from CGT-Prod using its cached TGT (round 2) and obtains a CGT, which it also caches. A CGT authorizes access to one or more namespaces belonging to the same producer. Finally, a consumer requests content from the producer using the corresponding CGT (round 3). TGT and CGT-s remain valid and re-usable until their expiration time runs out. Note that each round (1, 2 and 3) is realized as a single interest-content exchange.

Subsequent requests from the namespace(s) specified in the CGT require no involvement of either *TGT-Prod* or *CGT-Prod*. A consumer retrieves another content by directly issuing an interest containing the cached CGT. To access content under a different namespace, a consumer uses its cached TGT to contact CGT-Prod and request a new CGT.

For authentication and authorization, KRB-CCN must ensure that TGT-s and CGT-s issued to a specific consumer C_r are unforgeable, and not usable by clients other than C_r. Moreover, it must make sure that content authorized for C_r can only be decrypted by C_r. In the rest of this section we go into the details of how KRB-CCN achieves these requirements and functionalities.

4.2 Namespace-Based AC Policies

Instead of traditional service principals in Kerberos, KRB-CCN AC policies refer to namespaces, i.e, prefixes of content names that correspond to a producer. Recall that a content name is a URI-like string composed of arbitrary number of elastic name segments, separated by a '/' character. For example, consider a content named:

`/edu/uni-X/ics/cs/students/alice/images/img1.png`

The leftmost part, /edu/uni-X/ics/cs, defines this content's original producer's location. Subsequent name segments get increasingly specific, defining, e.g., location of the content in a directory structure on the producer.

KRB-CCN leverages this hierarchical name structure to implement AC policies based on content prefixes. For example, to grant Alice permission to retrieve contents under the prefix /edu/uni-X/ics/cs/students/alice, namespace /edu/uni-X/ics/cs/students/alice/* must be included under Alice's ID in the AC Policy Store, as shown in Figure 1. This entry would allow Alice

to retrieve `/edu/uni-X/ics/cs/students/alice/images/img1.png`, as well as any content with that same prefix.

Suppose that Bob is a faculty member of the faculty in the same institution and has privileges to retrieve contents under own (Bob's) private directory and any content of students' directories. Bob's entry in the AC Policy store would include two namespaces:

`/edu/uni-X/ics/cs/faculty/bob/*` and `/edu/uni-X/ics/cs/students/*`

The former allows Bob to access its own private directory under the faculty directory, but no other faculty's private directories. The latter allows Bob to access contents of any student directory under `/edu/uni-X/ics/cs/students/*`. Finally, suppose that Carl is a system administrator. As such, he has access to all content. Carl's entry in the AC Policy Store would be the namespace `/edu/uni-X/ics/*`, allowing access to any content with a name starting with this prefix; this includes all faculty and students' content.

If Alice (who is not yet "logged in", i.e., has no current TGT) wants to issue an interest for `/edu/uni-X/ics/cs/students/alice/images/img1.png` she first authenticates to `TGT-Prod` to get a TGT. Alice then uses the TGT to request a CGT from `CGT-Prod` for namespace `/edu/uni-X/ics/cs/students/alice/*`. Notice that Alice does not need to specify the actual content name – only the namespace. Therefore, `CGT-Prod` does not learn which content Alice wants to retrieve, only the producer's name. Since CGT is associated with `/edu/uni-X/ics/cs/students/alice/*`, it can be used for future interests within the same namespace, e.g., `/edu/uni-X/ics/cs/students/alice/docs/paper.pdf`.

4.3 Protocol

To retrieve protected content, C_r must go through all of `KRB-CCN`'s three phases, in sequence: authentication, authorization, and content retrieval. As discuss below, transition between phases is automated on the consumer side, i.e., it requires no extra actions. Table 1 summarizes our notation.

Authentication:
The first phase on `KRB-CCN` verifies consumer identity via authentication. The authentication protocol in Figure 2 is executed between C_r and `TGT-Prod`. If it succeeds, C_r receives a TGT, used in the authorization phase, as proof that C_r's identity has been recently verified.

C_r starts by issuing an interest with TGT suffix in the content name (e.g., /uni-X/ics/TGT). This interest carries as payload consumer's UID, i.e, C_r's username. Hence, the actual interest name also contains a hash of the payload as its suffix [1]. The interest is routed by CCN towards `TGT-Prod`. Upon the interest, `TGT-Prod` looks up UID in its user database and retrieves the corresponding public key. The protocol assumes that, when a user enrolls in the system, a public/private key-pair is generated. Alternatively, a password can be used for

[1] In CCN design, an interest carrying a payload must have the hash of the payload appended to its name.

Table 1. Notation summary

Notation	Description
N	A namespace prefix (e.g., edu/uni-X/ics/alice/)
C_r	Consumer
TGT_Name	Ticket-granting ticket name (e.g., edu/uni-X/ics/TGT) that will be routed towards `TGT-Prod`
CGT_Name	Content-granting ticket name (e.g., edu/uni-X/ics/CGT) that will be routed towards `CGT-Prod`
sk_C	Consumer Secret Key
pk_C	Consumer Public Key, including public UID and certificate
k_A	Long-term symmetric key shared between `TGT-Prod` and `CGT-Prod`
k_P	Long-term symmetric key shared between `CGT-Prod` and a given Content Producer
$s \leftarrow_{\$} \{0,1\}^{\lambda}$	Random λ-bits number generation
$ct \leftarrow Enc_k(pt)$	Authenticated Encryption of pt using symmetric key k
$pt \leftarrow Dec_k(ct)$	Decryption of ct using symmetric key k
$ct \leftarrow Enc_{pk}(pt)$	Authenticated encryption of pt using public key pk
$pt \leftarrow Dec_{sk}(ct)$	Decryption of ct using secret key sk

the same purpose, as discussed later. Once `TGT-Prod` successfully locates the user and retrieves the public-key, it proceeds with TGT generation. Otherwise, it replies with a special error content message indicating unknown user.

TGT is an encrypted structure with three fields: UID, k_{CGT}, and expiration date t_1. It is encrypted using k_A – a long-term symmetric key shared between `TGT-Prod` and `CGT-Prod`. Only `CGT-Prod` can decrypt and access cleartext fields of a TGT. Since C_r needs to present the TGT to `CGT-Prod` during the authorization phase, UID binds the TGT to C_r. This same UID is used later for namespace access rights verification. `CGT-Prod` uses t_1 to verify whether a TGT is still valid. TGT expiration time is a realm-specific (and usually realm-wide) parameter reflecting the duration of a typical user authenticated session, e.g., 8 hours. After TGT expires, C_r needs to repeat the authentication protocol with `TGT-Prod`. A TGT also contains a short-term symmetric key k_{CGT}, encrypted separately for `CGT-Prod` and C_r. The purpose of k_{CGT} is to allow C_r and `CGT-Prod` to communicate securely in the subsequent authorization protocol phase. In addition to the TGT, `TGT-Prod` generates $token^C_{CGT}$, which contains the same t_1 and k_{CGT} encrypted with the pk_C associated with UID.

To transmit the TGT to C_r, `TGT-Prod` responds with a content message containing the TGT and $token^C_{CGT}$, which is routed by CCN back to C_r. C_r cannot decrypt, access, or modify the TGT due to the use of authenticated encryption. C_r decrypts $token^C_{CGT}$ and caches the TGT for the duration of t_1, along with k_{CGT}. The TGT is presented to `CGT-Prod` every time C_r needs to request au-

thorization for a new namespace.

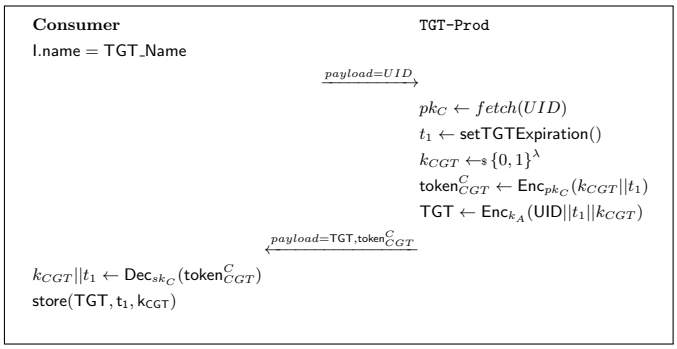


Fig. 2. Consumer authentication protocol

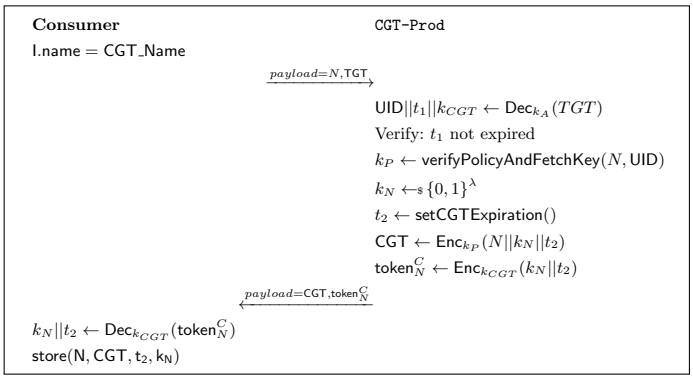


Fig. 3. Consumer-data authorization protocol

Authorization:
The authorization phase (Figure 3) is executed between C_r and `CGT-Prod`. It requires C_r to have a valid TGT, acquired from the authentication phase described above. Upon successful completion of the authorization protocol, C_r obtains a namespace-specific CGT, which demonstrates C_r's authorization to access a particular restricted content namespace. However, the CGT does not reveal C_r's identity to the content producer; C_r's authorization is ascertained based on possession of a correct session key.

C_r starts the protocol by sending an interest with name set to *CGT_Name*. The payload includes the namespace prefix N (e.g., `/edu/uni-X/ics/cs/students/alice/*`) for which authorization is being requested and a non-expired TGT. Optionally, if confidentiality for namespace N is an issue, C_r can compute $Enc_{k_{CGT}}(N)$. instead of sending N in clear. When `CGT-Prod` receives this interest, it uses k_A (long-term symmetric key shared by `TGT-Prod` and `CGT-Prod`) to decrypt the TGT and obtains UID, t_1 and k_{CGT}. Next, `CGT-Prod` checks TGT for expiration. It then optionally (if encryption was used in the interest) computes $N \leftarrow Dec_{k_{CGT}}(Enc_{k_{CGT}}(N))$.

If the TGT is successfully verified, `CGT-Prod` invokes *verifyPolicyAndFetchKey* procedure, which (1) fetches AC rules for user UID; (2) verifies if N is an authorized prefix for UID; and (3) returns k_P – symmetric key associated with the producer for N. k_P is later used to encrypt the CGT such that only the appropriate producer can decrypt it.

Similar to a TGT, a CGT carries an expiration t_2 and a fresh key k_N. The latter is used between C_r and the content producer for confidentiality and mutual authentication, as discussed later. However, instead of UID, a CGT includes N, i.e., a CGT proves to the content producer that whoever possesses k_N is authorized to access content under N. Also, a $token_N^C \leftarrow Enc_{k_{CGT}}(k_N||t_2)$ is sent to C_r, such that C_r can obtain k_N and t_2.

In response to a CGT interest, C_r receives a content packet containing the CGT and $token_N^C$. C_r decrypts $token_N^C$ using K_{CGT} and creates a cache entry containing: N, the CGT, k_N, and t_2. This cached information is used (until time t_2) in all future requests for content under N.

Authorized Content Request:
On the consumer (client) side, a `KRB-CCN` content request is similar to a regular CCN interest, except that C_r needs to include a valid CGT in the payload. An authorized interest name has the format: $N||suffix$ (e.g., `/edu/uni-X/ics/cs/students/alice/images/img1.png`), where N is authorized by the CGT, and $suffix$ specifies which content is being requested under namespace N. Note that, as long as C_r has proper access rights, a single CGT allows accessing any content with prefix N.

The secure content retrieval phase is in Figure 4. When the producer receives an interest for a restricted content, it first decrypts the CGT and verifies its expiration. Note that k_P used to decrypt the CGT is shared between the producer and `CGT-Prod`. Thus, successful decryption (recall that we use authenticated encryption) implies that CGT was indeed generated by `CGT-Prod` and has not been modified. The producer obtains k_N, which is also known to C_r. The producer encrypts requested content using k_N, i.e., $D' \leftarrow Enc_{k_N}(D)$. D' is returned to C_r, which decrypts it to obtain D.

Note that, by replaying the interest issued by C_r, anyone can retrieve D'. This might not appear problematic since only the authorized consumer (who has k_N) can decrypt D'. However, in some application scenarios this might be troublesome, e.g.:

– Production of content requires a lot of computation, e.g., expensive encryption. In this case, an adversary can replay legitimate interests previously issued by authorized consumers. The adversary's goal might be to mount a DoS attack on the producer.
– The producer might be a peripheral device, e.g., a printer. In this setting, the interest might be a request to print a (perhaps very large) document and returned content D might be a mere confirmation of it having been printed. In this case, the replay attack allows the adversary to print the same document multiple times, resulting in DoS.

This issue occurs since the producer does not authenticate C_r for each interest. A modified version of the protocol, shown in Figure 5, addresses the problem. It uses a challenge-response protocol that allows the producer to confirm that C_r possesses k_N before producing the content or providing service. As a down-side, this incurs an additional round of communication for the challenge-response protocol.

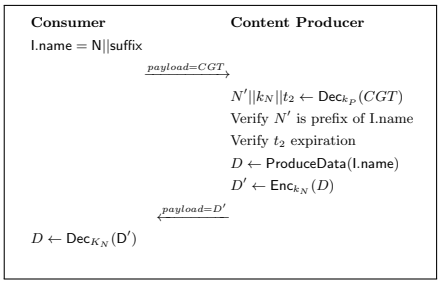


Fig. 4. Content retrieval *without* optional challenge-response based consumer authentication

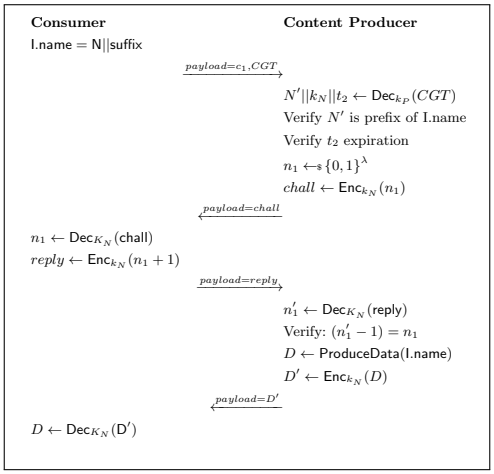


Fig. 5. Content retrieval **including** optional challenge-response based consumer authentication

Transparent Execution & Ticket Caching:
Recall that C_r must issue three types of interests, for: authentication, authorization, and the actual content request. This process is transparent to the user since `KRB-CCN` consumer-/client-side code handles these steps by following the work-flow in Figure 6.

Whenever C_r issues an interest, `KRB-CCN` client intervenes and checks whether the name is part of any restricted namespace. If so, it looks up the local cache of CGT-s to find a CGT for prefix N. If a valid CGT is found, it is added to

the interest payload and the interest is issued. A cached and valid CGT can be used to skip the first two phases, allowing authenticated and authorized content retrieval in one round.

If no valid cached CGT is found, KRB-CCN client looks up a cached TGT. If a valid TGT is found, the authentication phase is skipped. The client requests a CGT and uses it to request the actual content. This process takes two rounds.

In the worst case all three phases are executed, which results in three rounds of communication. Since consumers only request TGT and/or CGT-s when these tickets expire, ticket caching also reduces the number of requests (and overall traffic volume) flowing to TGT-Prod and CGT-Prod. In practice, we expect CGT-s and TGT-s to be long-lived, i.e., on the order of hours or days, similar to current single sign-on systems. This means that the bulk of authorized content retrieval can be performed in one round. If mutual authentication (per protocol in Figure 5) is demanded by the producer, one extra round is required.

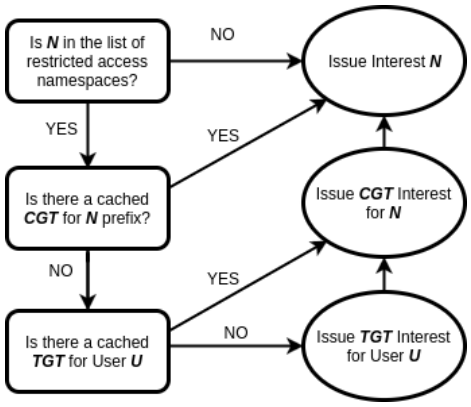


Fig. 6. KRB-CCN work-flow for transparent execution on consumers

5 Implementation & Performance Evaluation

This section discusses our KRB-CCN prototype implementation and its performance.

5.1 Methodology

KRB-CCN is implemented as an application service running as specific purpose producers that produce tickets. Also, consumer-side code is modified to implement the work-flow for authenticated and authorized content request in Figure 6. Our implementation uses the CCNx software stack [14] and the cryptographic

library Sodium [15]. Both publicly available and written in `C`. For authenticated PKE operations, we use Sodium Sealed-Boxes [16], implemented over X25519 elliptic curves. AES256-GCM [17] is used to encrypt-then-MAC, i.e., for authenticated symmetric-key encryption.

Experiments presented in this section were ran on an Intel Core i7-3770 octa-core CPU @3.40GHz, with 16GB of RAM, running Linux (Ubuntu 14.04LTS). Content payload sizes for interests were set to 10 kilobytes. Payload sizes of TGT and CGT contents are 228 bytes and 165 bytes, respectively. Each carries the respective ticket/token pair, as described in Section 4. In every experiment, each participating entity's process was assigned as a high priority, and each ran in a single processor core. Unless stated otherwise, results are an average of 10 executions, presented with 95% confidence intervals.

Figure 7 presents our network testbed. The goal is to evaluate `KRB-CCN`'s overhead. To avoid topology-specific delays, we used a minimal setup containing a single producer P, `TGT-Prod`, and `CGT-Prod`. These entities are interconnected by an unmodified CCNx Athena router.

5.2 Experiments

We start by measuring per-request processing times at each producer: `TGT-Prod`, `CGT-Prod`, and P. Each of these processes was executed $1,000$ times. Figure 8 presents the distribution, as box-plots, of processing time for verifying an incoming interest and replying with the content (either ticket, or authorized encrypted content) at each producer type. Figure 8 shows that the most computationally expensive part is TGT issuance (about $500\mu\ s$ per request). Higher computational overhead for TGT issuance makes sense because the authentication token ($token^{C}_{CGT}$ in Figure 2) is encrypted with C_r's public key. In case of password-based authentication, public key encryption is replaced by much faster symmetric key encryption using a password-derived key. This incurs much lower computational overhead on `TGT-Prod`.

Times for CGT issuance and content production are around $200\mu\ s$ and $300\mu\ s$, respectively. Time is naturally higher for the latter, since encrypted data is larger. In case of content production, the whole content (10kB) is encrypted. In a CGT request, only the CGT and the token need to be encrypted, resulting in a faster processing time.

To investigate how `KRB-CCN` entities cope under increasing congestion, we flood them with a massive number of simultaneous interests: from 300 to 3000. We then measure average Round-Trip Time (RTT) per type of issued interest. Figure 9 shows the RTTs for each response type. We also include the RTT for regular CCN content retrieval. Since it incurs no extra processing overhead, the regular content RTT is the lower bound for RTTs in CCNx implementation.

The average RTT for interests for TGT, CGT, and authorized encrypted content are similar. The latter is slightly higher as more data (10kB per interest) must traverse the reverse path back to the consumer. `KRB-CCN` requests incur in average $\sim 60\%$ higher RTT than unmodified content retrieval. In the largest

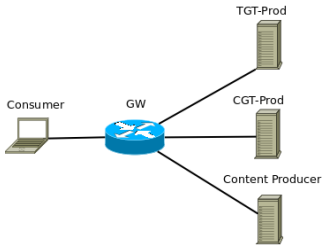


Fig. 7. Experimental testbed

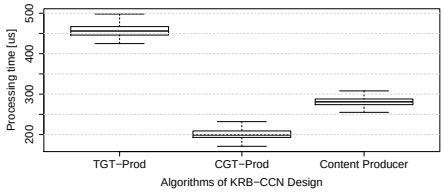


Fig. 8. Statistical distribution of the per-interest processing time (in μs) at each of KRB-CCN producers

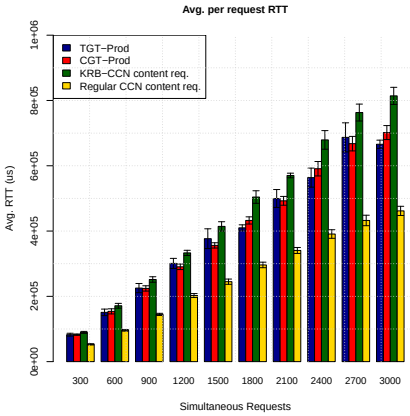


Fig. 9. Average RTT per request with massive simultaneous requests to the same producer

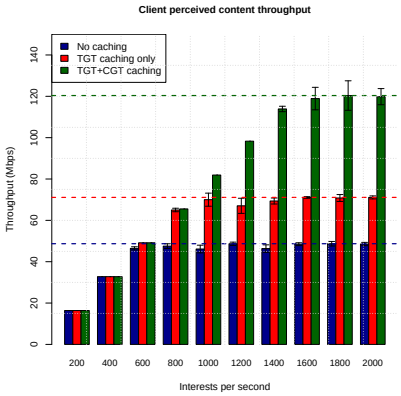


Fig. 10. Consumer perceived throughput under different ticket caching policies

scale test case, with 3000 simultaneous interests issued for each producer, content replies are received in less than $800ms$.

Finally, we also measure the overall throughput perceived by the consumer in three possible scenarios:

- **Cached TGT and CGT:** in this case the consumer requests contents under the same namespace. Therefore, the same (non-expired) cached CGT can be used for authorized content retrieval, allowing the client to skip the authentication and authorization phases.
- **Cached TGT only:** this is the case in which no AC ticket caching happens. It happens when the consumer always requests contents under different namespaces or because the realm owner demands consumers to request a fresh CGT for each content. In this case only the authentication part of the protocol is skipped at each request.
- **No caching:** This is the case in which the realm owner does not allow single sign-on through TGT caching nor authorization reuse through CGT caching.

Three requests (authentication, authorization, and content retrieval) are required for each content packet.

Recall that issuance of appropriate interests in each of the cases is automatically handled by `KRB-CCN` client software running on the consumer.

For each of the above cases, we gradually increase the rate of interests requested per second until throughput reaches its asymptotic limit. By analyzing throughput results, presented in Figure 10, we can observe the benefit of ticket caching. When both TGT and CGT caching are enabled, the client perceived throughput is higher than in the other cases, as actual content can be retrieved with a single interest. Conversely, caching only TGT-s is still better than not caching any type of ticket, because in this case the authentication phase can be skipped.

6 Related Work

Previous related efforts provide other types of security services currently available in IP-based networks. ANDaNA [18] is an anonymity service analogous to Tor [19] that uses CCN application-layer onion routing. Mosko, et al. [20] proposed a TLS-like key exchange protocol for building secure sessions for ephemeral end-to-end security in CCN. Similar to IPSec-based VPNs [21], CCVPN [22] is a network-layer approach for providing content confidentiality and interest name privacy via secure tunnels between physically separated private networks.

There are several CCN-based techniques that implement so-called Content-Based Access Control (CBAC). They tend to rely on content encryption under appropriate consumer keys to enforce AC. A group-based AC scheme for early versions of CCN was proposed in [3]. Policies tie groups of consumers to content, ensuring that only consumers belonging to authorized groups can decrypt restricted content. Similarly, Misra et al. [4] proposed an AC scheme based on broadcast encryption [23,24]. Wood et al. [5] proposed several AC schemes based on proxy re-encryption [25,26] to enable consumer personalized content caching. An attribute-based AC system, using attribute-based cryptography [27, 28] was proposed in [6]. CCN-AC [7] is a framework that unifies CBAC-type methods by providing a flexible encryption-based AC framework. It relies on manifest-based content retrieval specification (defined in CCNx 1.0 [29]) to enable flexible and extensible AC policy specification and enforcement. A similar approach is proposed in NDN-NBAC [8] framework. In these frameworks, data owners generate and distribute private keys to consumers via out-of-band channels. Producers receive corresponding public keys also via out-of-band channels. These public keys are used to encrypt one-time (per-content) symmetric keys.

In a different vein, Ghali et al. [9] proposed an Interest-Based Access Control (IBAC) scheme, wherein access to protected content is enforced by making names secret and unpredictable – based on encryption with keys known only to authorized consumers. Compared with CBAC, IBAC has the advantage of preserving interest name privacy and allowing content caching. However, IBAC

must be used in conjunction with CBAC to preclude unauthorized content retrieval via replay of previously issued obfuscated interest names.

In all schemes discussed above, authentication, authorization/AC, and confidentiality are often convoluted. In particular, producers are assumed to be implicitly responsible for authentication and authorization. This implies dealing with identity management and thus violating consumer privacy. Moreover, authentication and AC are enforced on a per-content basis which is unscalable and expensive. To the best of our knowledge KRB-CCN is the first comprehensive approach to address these issues by (1) separating authentication, authorization and content production among distinct entities; and (2) issuing re-usable authentication and authorization tickets for restricted namespaces.

7 Conclusions

We presented KRB-CCN – a comprehensive design for handling authentication, authorization, and access control in private CCN networks, while preserving consumer privacy. KRB-CCN is transparent to consumers and incurs fairly low overhead. We analyzed KRB-CCN security and assessed its performance based on a prototype implementation. Experimental results show that KRB-CCN is a practical and efficient means of providing multiple security services in private (stub AS) CCNs.

References

1. V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.
2. L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named data networking (ndn) project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.
3. D. K. Smetters, P. Golle, and J. Thornton, "Ccnx access control specifications," PARC, Tech. Rep, Tech. Rep., 2010.
4. S. Misra, R. Tourani, and N. E. Majd, "Secure content delivery in information-centric networks: Design, implementation, and analyses," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*. ACM, 2013, pp. 73–78.
5. C. A. Wood and E. Uzun, "Flexible end-to-end content security in ccn," in *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*. IEEE, 2014, pp. 858–865.
6. M. Ion, J. Zhang, and E. M. Schooler, "Toward content-centric privacy in icn: Attribute-based encryption and routing," in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*. ACM, 2013, pp. 39–40.
7. J. Kuriharay, E. Uzun, and C. A. Wood, "An encryption-based access control framework for content-centric networking," in *IFIP Networking Conference (IFIP Networking), 2015*. IEEE, 2015, pp. 1–9.

8. Y. Yu, A. Afanasyev, and L. Zhang, "Name-based access control," *Named Data Networking Project, Technical Report NDN-0034*, 2015.
9. C. Ghali, M. A. Schlosberg, G. Tsudik, and C. A. Wood, "Interest-based access control for content centric networks," in *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 2015, pp. 147–156.
10. B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications magazine*, vol. 32, no. 9, pp. 33–38, 1994.
11. M. Mosko, I. Solis, and C. Wood, "CCNx semantics," *IRTF Draft, Palo Alto Research Center, Inc*, 2016.
12. F. Ricciardi, "Kerberos protocol tutorial," *The National Institute of Nuclear Physics Computing and Network Services, LECCE, Italy*, 2007.
13. P. V. Mockapetris, "Domain names-concepts and facilities," 1987.
14. PARC, "Ccnx distillery," https://github.com/parc/CCNx_Distillery, 2016.
15. Sodium, "The sodium crypto library (libsodium)," https://github.com/jedisct1/libsodium, 2017.
16. D. J. Bernstein, "Curve25519: new diffie-hellman speed records," in *International Workshop on Public Key Cryptography*. Springer, 2006, pp. 207–228.
17. M. Dworkin, *Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC*. US Department of Commerce, National Institute of Standards and Technology, 2007.
18. S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun, "Andana: Anonymous named data networking application," *arXiv preprint arXiv:1112.2205*, 2011.
19. R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, Tech. Rep., 2004.
20. M. Mosko, E. Uzun, and C. A. Wood, "Mobile sessions in content-centric networks," in *IFIP Networking*, 2017.
21. N. Doraswamy and D. Harkins, *IPSec: the new security standard for the Internet, intranets, and virtual private networks*. Prentice Hall Professional, 2003.
22. I. O. Nunes, G. Tsudik, and C. A. Wood, "Namespace tunnels in content-centric networks," in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. IEEE, 2017, pp. 35–42.
23. A. Fiat and M. Naor, "Broadcast encryption," in *Annual International Cryptology Conference*. Springer, 1993, pp. 480–491.
24. D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Crypto*, vol. 3621. Springer, 2005, pp. 258–275.
25. G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.
26. R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 185–194.
27. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.
28. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 321–334.
29. I. Solis and G. Scott, "Ccn 1.0 (tutorial)," *ACM ICN*, 2014.

Appendix A: KRB-CCN Security Analysis

We now discuss security of KRB-CCN design presented in Section 4. We start by introducing the assumed adversary model and then split security analysis into 3 parts: user authentication, authorization, and content.

7.1 Adversary Model

We consider the worst-case scenario: C_r, that does not have a valid CGT or TGT, wants to fetch certain content. Thus, C_r must engage in KRB-CCN authentication (Figure 2), authorization (Figure 3), and authorized content retrieval (Figure 4 or 5, depending on the producers' requirements). TGT-Prod and CGT-Prod are trusted parties.

Adversary Goals and Capabilities: The adversary succeeds if it: **(1)** retrieves and decrypts unauthorized content; **(2)** retrieves any session key (k_{CGT}, k_N) or long-term key (k_A, k_P); or **(3)** forges tickets (TGT or CGT), tokens, or contents, leading C_r to believe that such forgeries were generated by honest parties; We consider a generic adversary *Adv* that is ubiquitous within a given KRB-CCN realm. *Adv* can perform the following actions:

- **Compromise and deploy compromised routers:** We allow *Adv* to compromise any number of existing CCN routers on the path between communication end-points. Hence, *Adv* learns all information in the routers and can change it at will. This includes changing FIB, PITs and content caches. We also allow *Adv* to deploy its own compromised routers.
- **Eavesdrop, analyze, and replay messages:** *Adv* can observe all CCN messages. Moreover, *Adv* can record and replay any traffic at will.
- **Issue interests and publish content:** *Adv* can deploy its own malicious consumers and producers. Thus, *Adv* can issue arbitrary interests and produce arbitrary content under its owned namespace.

Note: Protecting software and hardware of consumers or producers is out of scope for KRB-CCN. In particular, hardware and software exploits or malware are not considered in our analysis. Similar to IP-based Kerberos, KRB-CCN is an authentication and authorization system for secure computing platforms communicating over an untrusted (CCN) network.

We analyze KRB-CCN security in a top-down fashion. Security of content depends on security of the authorization phase, which, in turn, relies on secure user authentication. Therefore, we start by arguing that content retrieval is secure as long as authorization is secure. Then, we show that authorization is secure if user authentication is secure. Finally, we argue security of authentication by showing that only the owner of a certain user identity can gain access to content by claiming such identity. These notions are formalized bellow.

7.2 Content Retrieval Security

Content retrieval security means guaranteeing Property 1 (below) in case of content retrieval without consumer authentication. When consumer authentication is required, secure content retrieval implies both Properties 1 and 2.

Property 1 *Given an interest containing a CGT in the format* $CGT \leftarrow Enc_{k_P}(N||k_N||t_2)$ *as its payload, assuming that* k_N *is only known by* C_r *and* k_P *is only known by producer* Pr*, it must hold that:*

1. P_r *only replies to valid (non-expired and authentic) CGT-s issued by* `CGT-Prod` *and only* P_r *can decrypt such CGT-s;*
2. *A valid CGT for* N *can not be used to retrieve contents that do not belong to namespace* N*;*
3. *Only* C_r *can decrypt content generated by* P_r *in response to an interest containing such CGT;*
4. C_r *can check content authenticity (i.e., whether content originated at* P_r*) and integrity.*

Claim 1 `KRB-CCN` *content retrieval protocol (Figure 4) retains Property 1.*

Proof (sketch) 1 *Item (1) is assured because only* `CGT-Prod` *knows* k_P *and authenticated encryption with* k_P *is used to encrypt CGT. Moreover, the expiration* t_2 *is checked before issuing content replies. Item (2) follows from the integrity of received CGT (guaranteed by item 1) and the fact that* P_r *checks if the received request is for content that has the prefix* N*. Item (3) is true because content replies are encrypted under* k_N*, which is only known by* C_r*. Item (4) is assured by the use of authenticated encryption using* k_N*. Only* P_r *has* k_N*, because* k_N *is transmitted inside CGT, which is encrypted with* k_P*. Therefore,* C_r *can be sure that such content was in fact generated by* P_r *and also verify the received content integrity.*

Property 2 *Given an interest containing a CGT in the format* $CGT \leftarrow Enc_{k_P}(N||k_N||t_2)$ *as its payload, if* k_N *is only known by consumer* C_r *and* k_P *is only known by producer* Pr*, only* C_r *is able to use such CGT to get* P_r *to execute* $D \leftarrow$ ProduceData(I.name)*.*

Claim 2 `KRB-CCN` *content retrieval protocol with consumer authentication (Figure 5) retains Properties 1 and 2.*

Proof (sketch) 2 *The proof that content retrieval protocol with consumer authentication has Property 1 is equivalent to Proof 1. Property 2 is achieved because, in the protocol in Figure 5,* P_r *also plays a challenge response protocol with* C_r*, guaranteeing that* C_r *knows* K_N *before executing* $D \leftarrow$ ProduceData(I.name)*.*

Both properties assume that K_P is only known to `CGT-Prod` and P_r. This key is shared between them when P_r enrolls into a `KRB-CCN` realm. The properties also assume that K_N is only known to consumer C_r, which follows from authorization security, discussed in Section 7.3.

7.3 Authorization Security

Authorization security is represented by Property 3.

Property 3 *Given an authorization request interest for namespace N with $TGT \leftarrow \mathsf{Enc}_{k_A}(\mathsf{UID}||t_1||k_{CGT})$ as payload, assuming that k_{CGT} is only known by consumer C_r and k_A is shared between `CGT-Prod` and `TGT-Prod`, it must hold that:*

1. *`CGT-Prod` will only reply to valid (non-expired and authentic) TGT-s issued by `TGT-Prod` and only `CGT-Prod` is able to decrypt such TGT-s;*
2. *A CGT is only issued if the UID in the TGT has access to namespace N;*
3. *Only C_r is able to retrieve and verify integrity and authenticity of k_N;*

Claim 3 *`KRB-CCN` authorization protocol (Figure 3) retains Property 3.*

Proof (sketch) 3 *Item (1) follows from the use of authenticated encryption with k_A to encrypt the TGT and from the expiration verification of t_1. Recall that k_A is only shared between `CGT-Prod` and `TGT-Prod`. Item (2) is guaranteed by the integrity of received TGT (item (1)) and by the verification in the AC Policy database performed by `CGT-Prod`. Item (3) holds because k_N is encrypted with k_{CGT}, generating $token_N^C$ (and k_{CGT} is only known by C_r). Therefore, only C_r can decrypt $token_N^C$ and obtain k_N. Integrity of k_N is verifiable by C_r due to the use of authenticated encryption to generate $token_N^C$.*

Property 3 ensures that CGT-s are only issued to authorized consumers because only those are able to use k_{CGT} to decrypt $token_N^C$ and retrieve k_N. The only missing link in `KRB-CCN` security is to ensure that if a given C_r has k_{CGT} such C_r is in fact the owner of UID identity in the TGT. This is discussed next, in Section 7.4.

7.4 Authentication Security

Authentication security relies on making sure that only the owner of a claimed UID is able to retrieve k_{CGT}, i.e., the key included in the TGT issued for UID. This is stated by the following property:

Property 4 *Given an identity represented by (UID, sk_C, pk_C), where pk_C is a public-key know to `TGT-Prod` and sk_C is the associated secret-key only know to C_r – the owner of UID. It must hold that , for an issued $\mathsf{TGT} \leftarrow \mathsf{Enc}_{k_A}(\mathsf{UID}||t_1||k_{CGT})$, only C_r is able to retrieve the key k_{CGT}. Moreover C_r can verify integrity and authenticity of k_{CGT}.*

Claim 4 *`KRB-CCN` authentication protocol (Figure 2) retains Property 4.*

Proof (sketch) 4 *Since token_{CGT}^C is encrypted using pk_C, only the owner of UID can decrypt it and recover k_{CGT}. Integrity and authenticity of k_{CGT} are verifiable due to the use of authenticated encryption.*

In the password-based version of KRB-CCN, instead of encrypting under a public key, TGT-Prod would generate $token^{C}_{CGT}$ by encrypting with a symmetric key derived from password. Presumably, only the user that owns UID would know that secret password and be able to generate the same key to decrypt $token^{C}_{CGT}$. This approach shares the same characteristics and challenges (e.g., password strength, dictionary attacks, password memorability) of any password-based authentication. Password-based authentication is an option in KRB-CCN design. However, discussing specific challenges of password based authentication, though interesting, is not in the scope of the present work.

7.5 Discussion

By retaining Properties 1, 2, 3, and 4, KRB-CCN ensures secure authentication, AC, content integrity, and content confidentiality. As KRB-CCN protocol runs on consumers and producers, compromised routers and/or eavesdroppers are not able to violate what is guaranteed by the aforementioned properties.

Replay attacks and spoofed messages do not allow *Adv* to retrieve (unencrypted) content. If the content production is heavyweight, DoS via replay can be ruled out by enforcing mandatory mutual authentication (protocol in Figure 5).

In addition to the security services discussed in this section, KRB-CCN preserves consumer privacy. Producers only need to verify CGT-s authenticity and integrity, instead of consumers' identities. CGT-s do not carry UIDs, but instead associated keys, allowing the system to remain secure while preserving privacy. CGT-s are issued for namespaces (and not for complete content names). Thus, not even CGT-Prod (which knows the consumer UID) is able to predict which content will be requested by a consumer after issuing a CGT.

With ticket (TGT and CGT) caching, authentication and authorization can be skipped for subsequent interest within the same namespace. Therefore, most of the times KRB-CCN only requires low-cost (symmetric key) cryptographic operations from consumers/producers.