icnrg
Internet-Draft
Intended status: Informational
Expires: March 16, 2018

C. Wood
University of California Irvine

September 12, 2017

# Content-Locked Encryption and Authentication of Nameless Objects
# draft-wood-icnrg-clean-01

Abstract

This document specifies CCNx CLEAN - content-locked encryption and authentication of nameless objects. CLEAN describes how to transparently encrypt content objects in FLIC Manifests [I-D.irtf-icnrg-flic]. Relevant decryption information is carried in native FLIC nodes, i.e., without any extensions or modifications to FLIC. CLEAN transparently encrypts public data and supports application-specific configuration for private data.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 16, 2018.

Table of Contents



## 1. Introduction

In CCN, nameless objects are content objects which do not carry a Name TLV field. Thus, a necessary requisite to retrieve them from the network is to know their respective ContentObjectHashRestriction, or ContentId. A ContentId is the cryptographic hash of a content object [I-D.irtf-icnrg-ccnxsemantics]. A router may only forward a nameless content object if its cryptographic hash digest matches the ContentId of the corresponding interest.

By definition, a consumer cannot (with overwhelming probability) request a nameless content object without knowledge of its ContentId. Manifests are network-level structures that convey ContentIds to consumers. FLIC Manifests [I-D.irtf-icnrg-flic] are one type of Manifest structure. Manifests typically group segments of a large piece of data under a common name. For example, suppose there exists a content with the name /foo/bar, which has a total size beyond the 64KB limit imposed by the CCN packet [I-D.irtf-icnrg-ccnxmessages]. The producer of /foo/bar can segment the data into fixed size chunks and, for each chunk, create a nameless content object whose payload is the chunk. Then, the producer may create a Manifest with the name /foo/bar which contains the references to each of these constituent nameless object parts. To fetch /foo/bar, a consumer then does the following:

1. Issue an interest for the name /foo/bar.

2. Receive, verify, and parse the Manifest.

3. Issue requests for each nameless content object using the provided ContentIds.

(See [I-D.irtf-icnrg-ccnxsemantics] for more details.)

By default, the data contained inside each nameless content object is unencrypted. If confidentiality is required, producers must explicitly encrypt data prior to FLIC encoding. This arrangement is not ideal. By default, all data should be encrypted, even if it is public. CLEAN - content-locked encryption and authentication of nameless objects - is a mechanism that achieves this goal. CLEAN builds on recent advances in message-locked encryption ([MLE]) to encrypt nameless objects by default without invalidating their natural de-duplication properties.

## 1.1. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The following terms are used:

Nameless object: A CCN content object packet which contain a Name TLV field.

CLEAN collection: A manifest tree encrypted via CLEAN.

## 2. CLEAN Crypto

CLEAN only relies on MLE, which is a form of encryption by which the encryption key for a message is derived from the message itself. For example, a message M may be encrypted by a key k = H(M), where H is a suitable cryptographic hash function for use in MLE constructions. (See [MLE] for more details.) The encryption of M is then computed as M' = Enc(k, M), where Enc is a symmetric-key encryption algorithm suitable for MLE.

MLE is deterministic. Identical messages will be encrypted to identical ciphertexts. As a result, MLE supports natural de-duplication of data based on ciphertext equality.

## 3. CLEAN Construction

Let D be a piece of data for which a producer P would normally create a Manifest with name N. Let C_1,...,C_n be n nameless content objects created from D. That is, each C_i contains D_i, the i-th

chunk of D. See [I-D.irtf-icnrg-flic] for more details about this chunking procedure.

CLEAN works as follows:

1. P computes k = KDF(ctx, H(D)), where KDF is any suitable key derivation function, e.g., HKDF [RFC5869], and ctx is an application context string for the KDF. By default, ctx is an empty string, meaning that k = KDF(H(D)).

2. For each C_i in C_1,...,C_n, P derives k_i = KDF(k || i) and uses it to compute C_i' = Enc(k, C_i). Encryption is only performed over the payload of the content, not the headers.

3. From C_1',...,C_n', P creates the manifest M(N) as described in [I-D.irtf-icnrg-flic].

4. P inserts H(D) into the root node of M(N). (This is described in Section Section 5.)

The context string "ctx" is used to scope the CLEAN encryption to an application-specific context. By default, there is no context, as data is assumed to be public. This means that different producers generating the same content with an empty context will create the same encrypted content objects and, potentially, the same manifest tree.

This may not always be desirable. To constrain the context of a CLEAN collection, applications may opt-in to CLEAN and specify encryption contexts. To prevent an attacker from hijacking interests for CLEAN objects, or from creating duplicate content objects, these contexts must be secret to the producer. We describe several candidate contexts below:

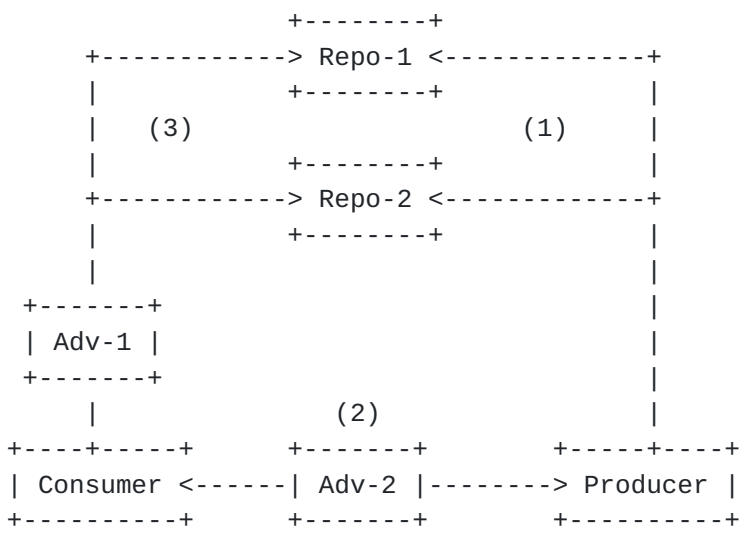
- H(r), where r is a random 32B string.

- H(sk), where sk is a long-term secret key kept by the producer.

The context string must be transferred to the consumer so as to derive the same encryption key. We describe how to do this in Section Section 5.

## 4. CLEAN Publishing and Fetching

There are at least three steps in the CLEAN publishing process, drawn below and labelled for clarity:




1) The producer creates the CLEAN collection and, optionally, uploads the CLEAN contents, without the root manifest, to dedicated content repositories. 2) A consumer fetches the root manifest from the producer. 3) A consumer proceeds to fetch the rest of the CLEAN collection from either the producer or the dedicated repositories.

If an eavesdropper only sees traffic from the consumer to the repository, then CLEAN keeps this data safe. If an eavesdropper can also see traffic from the consumer to the producer, then it can learn the necessary information required to decrypt the traffic. Therefore, to ensure safety, consumers SHOULD always fetch the root manifest over a secure session. We expand on this point in Section Section 6.1.

## 5. FLIC Support

Consumers require two pieces of information to decrypt a CLEAN collection: "H(D)" and "ctx". The [I-D.irtf-icnrg-flic] format already includes the metadata value OverallDataDigest. For a given FLIC node N, this value corresponds to H(D), where D is the contiguous set of application data in the nameless content objects contained in N. Carrying "ctx" requires an extension to FLIC with the type "clean_context".

## 6. Use Cases

This section describes how to use CLEAN to protect public and private data. Private data is that which must be kept confidential, i.e., it requires some form of access control. Public data can be freely accessed by anyone.

### 6.1. Public Data

Since public data requires no access control, applications need not provide a CLEAN context string. By definition, anyone should be able to access public data. CLEAN is still useful in this case since it requires an eavesdropper to fetch the root manifest in order to decrypt the leaves. Simply observing the request and response for an encrypted CLEAN object - not the root manifest - in transit does not reveal the necessary information to decrypt the response. Consumers may choose to fetch the root manifest over a secure session, such as that enabled by [I-D.wood-icnrg-ccnxkeyexchange] and [I-D.wood-icnrg-esic], to prevent leaking the necessary decryption information.

### 6.2. Private Data

Private data requires access control. In this case, applications MUST provide a secret context string to the CLEAN encryption algorithm. This prevents another (malicious) producer from generating the same set of CLEAN objects. Moreover, it must be assumed that all traffic can be observed in transit. Thus, the root manifest MUST be protected either at rest by encryption-based access control or in transit with a secure session, i.e., with [I-D.wood-icnrg-esic] bootstrapped by [I-D.wood-icnrg-ccnxkeyexchange].

## 7. Security Considerations

The CLEAN security model depends on the root manifest being protected either at rest or, optionally, in transit. If the root is protected at rest via some access control mechanism, then CLEAN remains secure in the MLE model. MLE security also holds if the root is encrypted only in transit over a secure session, i.e., with [I-D.wood-icnrg-esic] using a key bootstrapped by [I-D.wood-icnrg-ccnxkeyexchange]. See [TRAPS] for more details about this analysis.

## 8. Normative References

[I-D.irtf-icnrg-ccnxmessages]
Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", draft-irtf-icnrg-ccnxmessages-04 (work in progress), March 2017.

[I-D.irtf-icnrg-ccnxsemantics]
Mosko, M., Solis, I., and C. Wood, "CCNx Semantics", draft-irtf-icnrg-ccnxsemantics-04 (work in progress), March 2017.

[I-D.irtf-icnrg-flic]
Tschudin, C. and C. Wood, "File-Like ICN Collection (FLIC)", draft-irtf-icnrg-flic-00 (work in progress), June 2017.

[I-D.wood-icnrg-ccnxkeyexchange]
Mosko, M., Uzun, E., and C. Wood, "CCNx Key Exchange Protocol Version 1.0", draft-wood-icnrg-ccnxkeyexchange-02 (work in progress), July 2017.

[I-D.wood-icnrg-esic]
Mosko, M. and C. Wood, "Encrypted Sessions In CCNx (ESIC)", draft-wood-icnrg-esic-00 (work in progress), March 2017.

[MLE] Mihir Bellare, ., Sriram Keelveedhi, ., and . Thomas Ristenpart, "Message-locked encryption and secure deduplication", n.d., <https://eprint.iacr.org/2012/631.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[TRAPS] Wood, Christopher., "Protecting the Long Tail: Transparent Packet Security in Content-Centric Networks", n.d..

## Author's Address

Christopher A. Wood
University of California Irvine

EMail: woodc1@uci.edu