ICNRG Working Group
Internet-Draft
Intended status: Informational
Expires: March 16, 2018

M. Mosko
PARC, Inc.
C. Wood
University of California Irvine
September 12, 2017

Encrypted Sessions In CCNx (ESIC)
draft-wood-icnrg-esic-01

Abstract

This document describes how to transport CCNx packets inside an encrypted session between peers that share a traffic secret, such as that which is derived from [CCNxKE]. The peers create an outer naming context to identify the encryption session in one direction between the consumer and the producer. The consumer sends encrypted Interest messages to the producer, who responds with encrypted Content Objects. Inside the outer context, the consumer sends Interests with different names, which the producer may respond to or may send InterestReturns for. There does not need to be a naming relationship between the outer names and the inner names. The inner content is still protected by normal CCNx authentication mechanisms and possiby encrypted under other schemes.

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.

Copyright Notice

Table of Contents

1. Introduction

CCNx packets [MESSAGES] contain a fixed header, optional hop-by-hop headers, a CCNx Message, and a validation section. Encrypted Sessions in CCNx (ESIC) describes how to to transport encrypted CCNx packets inside other CCNx packets. The outer packet (the wrapper) uses a CCNx name that identifies the encrypted session while the

inner (encrypted) portion remains hidden and private to an outside observer.

ESIC defines a new field Encapsulated (T_ENCAP) that may occur in both an Interest (T_INTEREST) and Content Object (T_OBJECT). The T_ENCAP field contains the encryption of the inner CCNx Packet.

Because the use of an outer CCNxPacket, the total packet length of the inner CCNxPacket may need to be limited to less than the maximum of 64 KB. ESIC allows the use of a compressor before the encryptor, so it is likely that a packet that would overflow the 64 KB limit could be compressed by enough to allow for an outer CCNxPacket. This consideration for the PacketLength is separate from concerns about path MTU.

It is a requirement of ESIC that one inner packet fit in one outer packet. This is because ESIC does not define a method to issue extra outer interests to fetch extra outer content objects. It relies entirely on Interests generated by the consumer application.

ESIC defines a control channel within the outer context by using special names with the inner packets. These names allow signaling between the two encryption endpoints for features such as alerts and rekeying requests.

ESIC defines how to use a traffic secret (TS), such as derived from CCNxKE, to encrypt multiple packets in a consumer-producer session. Each direction will use separate derived keys. If one wishes to have a reverse traffic flow (interests from producer fetching content objects from the consumer), then one must share a second TS and use it with the roles reversed, but otherwise it works exactly as in the first case.

The mechanism by which this symmetric key is obtained is outside the scope of this document; These keys could be pre-shared or derived from an online key-exchange protocol [CCNxKE].

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:

- Inner Packet: A fully-formed CCNx packet (fixed header through validation) that is carried encrypted inside a T_ENCAP TLV.

- Outer Packet: A fully-formed CCNx packet that carries the outer context of an encrypted session.

- Outer Name: The name of the outer packet.

- Inner Name: The name of the inner packet (not visible in transport).

- Control channel: the use of Inner Packets to convey control signaling between encryption endpoints using a special Inner Name.

2. Stateless packet keys

ESIC assumes that the consumer and producer share a Traffic Secret (TS), usually derived as per CCNxKE. Regardless of how the TS is derived (TODO: it needs to meet some so-far unstated requirements), there are four secrets derived from the TS, as per CCNxKE Section 9.5. This specifies how to generate the Client Write Key, Server Write Key, Client Write IV, and Server Write IV.

The AEAD nonce (IV) is derived as specified in [TLS13]. In particular, the length of the IV for each AEAD operation is set to max(8 bytes, N_MAX), where N_MIN must be at least 8 bytes [RFC5116]. With this length, the nonce is initialized by:

1. Padding the 64-bit per-packet AEAD sequence number to the left with zeroes so that its length is equal to the IV length.

2. This padded sequence number is then XORed with the consumer or producer IV, depending on the role.

TODO: Should we allow CCNxKE to specify the starting chunk number so

it does not always start at 0? It would need to be encoded in the MoveToken.

3. Inner and Outer Contexts

The inner context is a CCNx packet with meaning to the consumer and producer. They may be clear text or they make use additional encryption, such as group keying, broadcast encryption, homomorphic encryption, or something else. The consumer sends an Interest packet with an Inner Name (plus other optional fields as normal) and expectes to get back a Content Object or InterestReturn packet with corresponding name and fields.

The outer context names the encryption session and sequences packets. ESIC does not expect a one-to-one correspondence of outer name and inner name. If a consumer, for example, sends 3 interests with outer names NO1, NO2, NO3 and inner names NI1, NI2, and NI3, the producer can return those names in any order. It could put content objects with name NI3 in NO1, NI1 in NO2, and NI2 in NO3. ESIC does expect

normal CCNx processing rules to be followed for the inner packets, therefore we would expect at most one inner packet returned for each inner Interest. That inner packet could be either a Content Object or Interest Return.

3.1. Outer Context Names

The outer context name is a routable prefix PREFIX followed by a session ID (SID) followed by a ChunkNumber (Chunk). The chunk number is a monotonically increasing number. The outer name is clear text, visible to all observers.

The PREFIX and SID are derived outside of ESIC. In normal use with CCNxKE, the PREFIX is either the same prefix as used in the key exchagne or it is derived from within CCNxKE from Prefix2 or the MoveToken. The SID is created by the producer and given to the consumer inside CCNxKE.

OuterName := ccnx:/PREFIX/SID=sid/CHUNK=chunk

Chunk numbers are limited to 8 bytes and do not wrap around. When the consumer gets near the end of the sequence number space, it must

request a re-keying via the control channel. Because CCNx in a pull-driven model, the consumer is reponsible for the chunk number and thus responsible for requesting the re-keying. The producer may also request a re-keying for its own reasons.

3.2. Outer Packet

The outer packet will have a Fixed Header, per hop headers, a CCNx Message with the Outer Name, and a Validation section (ValidationAlg and ValidationPayload). The outer packet is visible to 3rd parties in its entirety. Only the 'value' of T_ENCAP TLV field inside the CCNx Message is encrypted. The T_ENCAP TLV Value is the AEAD 'plaintext' that will be converted to the 'ciphertext'. In the outer packet, only the CCNx Message and the ValidationAlg are covered by the authentication token

The Outer Packet has a Validation section. The ValidationAlg will have a 0-length ValidationType of T_SESSION, which indidates that the encryption context must be derived from the SID in the name.

The Associated Data (in AEAD) covered by the validation output is from the beignning of the CCNx Message up to but not including the T_ENCAP Value concatendated with the ValidationAlg TLV. That is, it skips the T_ENCAP TLV Value.

The ValidationPayload contains the AEAD authentication token.

If the Producer cannot satisfy an Inner Packet Interest, it will encapsualte an InterestReturn inside an OuterPacket of PacketType ContentObject. That is, the InterestReturn is end-to-end signaling about the inner context.

If the Producer has an error with the Outer Context, it may return an InterestReturn for the outer context as normal for Interest processing.

3.2.1. Consumer Outer Packet

The outer packet from the consumer to the producer will always be of PacketType Interest. They may have any of the normal Interest per-hop headers (e.g. InterestLifetime), which will be visible to 3rd parties and not protected by the encryption or authentication.

The Outer Context has a T_INTEREST message type, which contains a T_NAME of the Outer Name. It may have other additional metadata in clear text. The T_INTEREST container is protected by the encryption authenticator. Finally, the T_INTEREST has a T_ENCAP field that contains the encryption of the Inner Packet. The encryption will use the algorithm negotiated as part of the SID (i.e. AES-GCM).

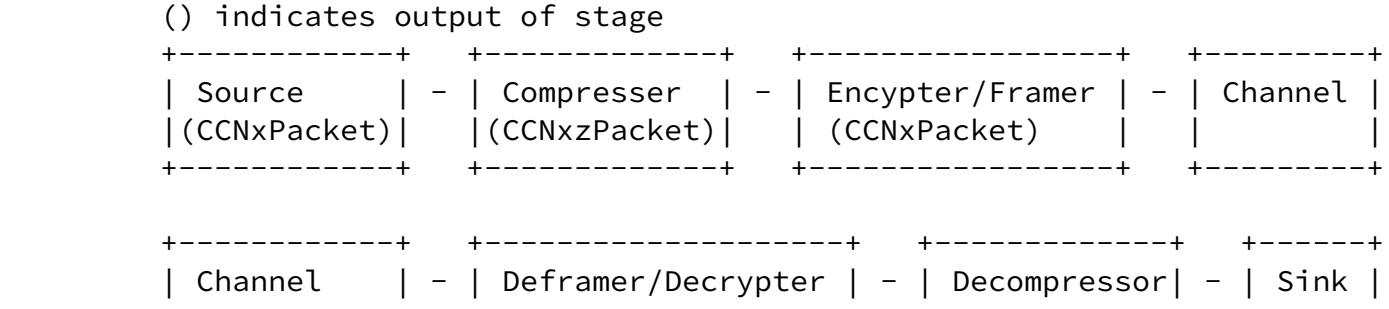
3.2.2. Producer Outer Packet

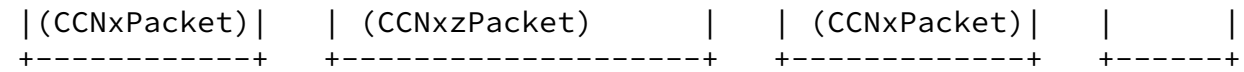
The producer will only send PacketType ContentObject back to the consumer. The Inner packet may be either an InterestReturn or a ContentObject corresponding to the Inner Packet interest.

The outer packet may have per-hop headers (e.g. RecommendedCacheTime) that affect the encrypted packet. These are independent from the inner Per Hop headers. The outer MessageType is always T_OBJECT. It may have normal metadata for a content object, such as ExpiryTime, which affect only the outer packet. Finally, it has a T_ENCAP that contains the wrapped inner Packet.

3.3. Processing Chain

The processing chain from the Source to the Sink is shown below. The compression/decompression stages are optional and are not strongly tied to the encrypted session. If used, we assume the compression protocol is session specific to avoid state snooping (e.g. such as in CRIME attack).





- Source: The source of an Inner Packet.
- Compressor: Optional component to reduce the size before encryption.
- Encrypter/Framer: Creates the ciphertext of the CCNx(z)packet to produce the T_ENCAP, constructs the outer packet, computes the authentication token and generates the ValidationPayload.
- Channel: Carries the wireformat outer packet
- Deframer/Decrypter: Verifies the authenticator, decrypts the T_ENCAP, and passes the Inner Packet to the Decompressor.
- Decompressor: Optional component to expand the inner packet
- Sink: The sink of an Inner Packet.

The Encrypter/Framer will generate outer names with sequential outer name chunk numbers.

The Deframer/Decryptor will extract the SID and chunk number from the outer name and use those to create the packet key (see below). Using the packet key, it will verify the authentication token and if successful decrypt the T_ENCAP. The output of the T_ENCAP will then be passed to the Sink.

3.4. Transport State Machine

ESIC uses a state machine to manage the ephemeral session such that the Producer knows when the Consumer is finished with the SID. It also will try to re-request packets that fail authentication before sending its own InterestReturn up the Sink.

The protocol begins with each side knowing the four keys (see Stateless Packet Keys below), the Session ID (SID), and the routable prefix PREFIX.

The receiving process uses a replay buffer to prevent replay attacks.

The buffer tracks the last N out-of-order verified chunks plus the cumulative verified chunk number. TODO: Sort this out how to avoid replay attacks without requiring reliable in-order delivery.

Protocol of Encrypter/Framer:

- Initialize: set NextChunkNumber = 0, State = Waiting
- Waiting: Wait for packet from Source (or compressor). On packet receive, State = Send
- Send:
  - Generate packet key for NextChunkNumber
  - Create outer packet with name /PREFIX/SID=sid/ CHUNK=NextChunkNumber and the input packet as cleartext in the T_ENCAP.
  - Run the AEAD scheme authenticating and encrypting. Note the prior description of the split Associated Data before and after the plaintext.
  - Increment NextChunkNumber
  - Send the packet
  - State = Waiting

Protocol of the Deframer/Decrypter:

- Initialize the replay buffer to empty, State = Waiting.
- Waiting: wait for packet, on input from channel State = Receive
- Receive:
  - Extract the SID and ChunkNumber from name
  - If replay, drop
  - Authenticate the packet
    - If failed on consumer, send InterestReturn to Source with "X Error" (TBD)

    + If failed on producer, send failure message to Sink so it can send end-to-end InterestReturn back over channel (if desired) with "Y Error" (TBD)

  * Add packet to replay buffer

  * Decrypt packet

  * Pass decrypted packet to Sink/Source (or decompressor)

4. Control Channel

The consumer and producer will need to exchange signaling about the encryption context. Control and data traffic should be indistinguishable to an external observer. Therefore, all control signaling is done within the same outer names as data traffic.

Control signaling is done with a normal Inner Packet that pushes data to the other side. We use an Interest with an Inner Name of the form shown below, where '_direction_' is 'up' from the consumer to producer or 'down' for the producer to consumer. This allows each side to maintain its own sequence number space in the 'seqnum'. This is similar to the use of the sequence number in the DTLS record layer.

Like DTLS, ESIC control messages are unreliable, though they are uniquely named.

The payload of the control Interest uses a TLV equivalent of the TLS record format for handshake and alert messages. Application data is never communicated in these records, as they use an Inner Packet with a different Inner Name. Inside the payload, a TLV type of Alert (21) or Handshake (22) indicates the purpose of the TLV value. One may concatenate multiple records in to one payload.

ControlName := ccnx:/localhost/esic/_direction_/SID=sid/SEQNUM=seqnum

4.1. ESIC Control Packets

A control packet is a CCNx Interest Inner Packet. The name of the control packet is as above in the /localhost/esic namesapce. The Payload of the Interest is the actual data.

The ESIC control packet SHOULD be padded out to a length that is indistinguishable from other traffic in the given _direction_.

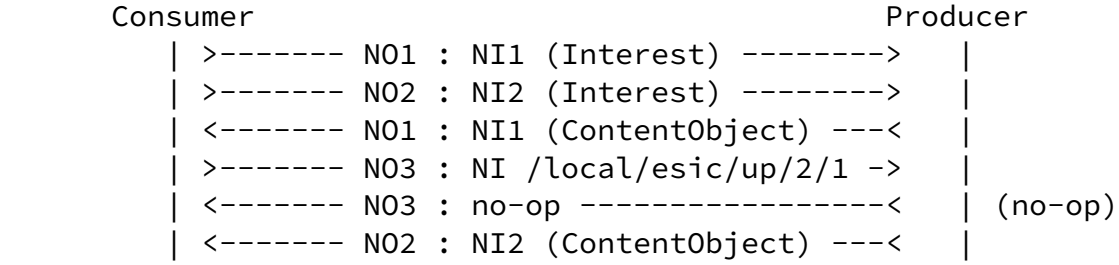
The Payload of the Interest contains a set of TLV records using the normal CCNx TLV encoding. The TLV types and values are defined in the next section.

In the 'up' direction from the consumer to the producer, a control packet can be inserted into the Interest stream as normal. The producer may use this extra outer name to return its own control message or send a "no-op" back to consume the extra name.

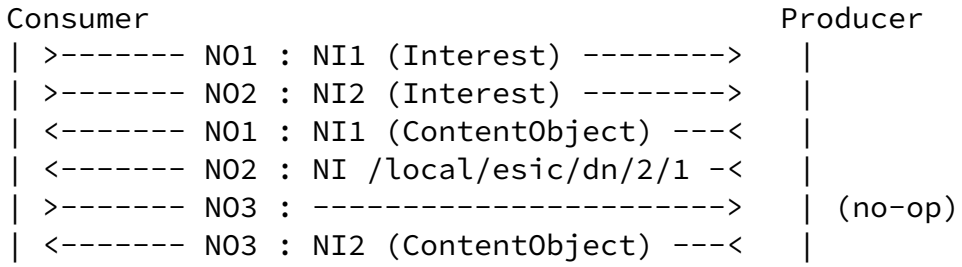
In the 'down' direction from the producer to the consumer, there is no pre-allocated outer name available. The producer can only send the consumer a control message if the consumer has outstanding Interests up to the producer. If there is one or more oustanding interests in the outer name space, the producer normally would send a Content Object or Interest Return corresponding to some inner name. In this case, the producer would instead inject a control packet Interest in the downstream. This means the producer is now short one outer Interest in the upstream direction. Therefore, whenever the Deframer/Decrypter sees a control message in the downstream direction, it MUST insert an upstream "no-op" packet, padded out to statistically undetectable length, to give the producer back a missing name slot.

We allow one ESIC control packet in one outer packet. However, we allow multiple Alert messages to be encoded in the payload, so long as it remains indistinguishable from other packets in the given _direction_.

Example from a consumer to a producer, where "NO" means "name outer" and "NI" means "name inner".



Here is an example from a producer to a consumer. The producer uses the second avaialble name NO2 to send a control message to the consumer. The consumer must then send a no-op packet back up to the producer so it can return the final data packet NI2 inside NO3.



TODO: Add examples with loss

4.2. ESIC Control Messages

ESIC adopts the TLS 1.3 Alert Protocol for its control messages. The TLV type of the message inside the control packet payload is taken from the enum AlertDescription. As per TLS 1.3, fatal Alert messages are an immediate close of the ESIC session.

As per TLS 1.3, each party MUST send a close_nofity message closing the write side of the connection. In ESIC, this means that when a consumer is done requesting data, it should send a final close_notify. The producer should then use this outer name to send back its own close_notify. If for some reason the producer must close before the consumer, it should inject its own close_notify discarding all remaining data and the consumer should send back upstream a close_notify.

The KeyUpdate messages function as per TLS 1.3 Sec 6.3.5.3. Either side may generate a KeyUpdate message and begin transmitting with the new key. The other side must update their own key and issue its own KeyUpdate message.

5. The ESIC API

In this section we describe the ESIC API. Before doing so, we highlight some details that molded the API for both consumers and consumers.

- Encrypted sessions are bound to names instead of addresses. Consequently, in addition to a set of trusted keys, sessions between a consumer and producer require only a name to be created.

- Sessions are created by an active consumer with a passive peer (producer). Thus, the API must reflect these roles.

- Consumers send and receive whole CCNx messages over a session. Thus, simple read and write functions must be exposed via the API.

- Sessions are not full duplex by default. A producer must specify in its ServerConfiguration construct that it wishes to send interests to the consumer. To maintain transparency, the modality of the resulting session is not reflected in the API.

These observations are distilled in the following ESIC API.

```
# @Consumer: create a secure session with a producer
CCNxSecureSession *ccnxSecureSession_Connect(CCNxPortal *portal,
    PARCIdentity *identity, CCNxName *servicePrefix);

# @Producer: create a passive listener
CCNxSecureSession *ccnxSecureSession_CreateServer(CCNxPortal *portal,
    CCNxKeyExchangeConfig *config, CCNxName *servicePrefix);

# @Producer: accept uni- and bi-directional sessions
CCNxSecureSession *ccnxSecureSession_AcceptConnection(CCNxSecureSession *sessio
CCNxSecureSession *ccnxSecureSession_AcceptBidirectionalConnection(CCNxSecureSe

# Send a CCNx message
# Override the outer name with the `response` parameter if needed
void ccnxSecureSession_SendMessage(CCNxSecureSession *session,
    CCNxTlvDictionary *message, const CCNxStackTimeout *timeout, CCNxName *resp

# Receive and decapsulate a CCNx message
```

```
# Store the outer name in the `response` parameter.
CCNxMetaMessage *ccnxSecureSession_ReceiveMessage(CCNxSecureSession *session,
    const CCNxStackTimeout *timeout, CCNxName **response);
```

6. Security Considerations

It may be possible for an observer to identify which outer packets contain a control (alert) message if the ACK response time shows significant statistical timing different from the normal flow of messages.

TODO.

7. References

7.1. Normative References

[CCNxKE] "CCNx Key Exchange Protocol Version 1.0", n.d., <https://github.com/parc/ccnx-keyexchange-rfc>.

[MESSAGES] "CCNx Messages in TLV Format", n.d., <https://tools.ietf.org/html/draft-irtf-icnrg-ccnxmessages-02>.

[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>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/info/rfc5116>.

[TLS13] RTFM, Inc, ., "The Transport Layer Security (TLS) Protocol Version 1.3", n.d., <https://tools.ietf.org/html/draft-ietf-tls-tls13-13>.

7.2. Informative References

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/info/rfc5288>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <https://www.rfc-editor.org/info/rfc5389>.

Appendix A. Test Vectors

A.1. Sample Encryption TLVs

TODO

A.2. Interest Encapsulation Examples

TODO

A.3. Content Object Encapsulation Examples

TODO

Authors' Addresses

Marc Mosko
PARC, Inc.

Email: marc.mosko@parc.com

Christopher A. Wood
University of California Irvine

Email: woodc1@uci.edu