File-Like ICN Collections (FLIC)

draft-irtf-icnrg-flic-04

IETF 115, London

Marc Mosko

PARC

Dave Oran

Network Systems Research & Design

Outline

- FLIC background and review
- A walk through of the main FLIC features.
- Known issues for -04

Background

- FLIC was first conceived back in the CCNx 1.0 days, maybe 2014. The -00 draft expired in 2017 and the -03 draft expired in May 2022. We are now on the -04 draft.
- Manifests are useful in NDN, but pretty much critical in CCNx:
  - "nameless objects" that just have a hash
  - segmentation for large objects
  - Collections, like directories of objects "lower" in a namespace
- It's been implemented and in use all along

Current State of Affairs

- Original authors have mostly moved on to other stuff
- DaveO cajoled Marc and Christian to resurrect work with informal meeting at ICNRG in Montreal.
- The -04 draft has cleaned up the draft. Thank you to Ken Calvert for his -03 comments.

What FLIC does

- It provides a manifest of hashes that make up all the segments of a piece of application data.
- The manifest is hierarchical – that is the hash pointers can point to application data or to more manifests.
- There is a canonical traversal order. Metadata could provide other traversal hints, such as for video.
- FLIC has its own, extensible, encryption mechanism. Manifest encryption does not need to be related to content encryption.
- FLIC has several Interest construction techniques. The publisher can choose one or more of these naming techniques. More techniques could be added.

As a Manifest

- In NDN and CCNx, it is most efficient to use hash-based names, as forwarders can match the name exactly.
- A manifest is one technique to distribute those hashes. These have been in use since the early CCNx 0.x days.
- Manifests can also offload authentication overhead, as one only needs to sign the root Manifest.
- A FLIC manifest has one or more Hash Groups. Each HG can have its own metadata and Name Constructor (in a few slides).

Traversal Order

- FLIC defines a specific traversal order (pre-order, i.e. top-to-bottom, left-to-right).
- A consumer can retrieve pieces in any order, but the traversal order defines the proper way to assemble the bytes to reconstruct the original data.
- Some uses (e.g. video) may not care about perfect reassembly and may use metadata hints to skip around.
- The hash pointers in a manifest can point to other manifests (M) or application data (D). Within a manifest, the publisher can organize pointers as desired, e.g. MMMDDD, DDMMM, DMDM, MMMM and DDDD, etc.

Metadata and Annotations

- Metadata is for the manifest overall or Hash Groups. It applies to a collection of hashes.
  - Manifest: Subtree Size, Subtree Digest, Locators
  - Hash Group: LeafSize, LeafDigest, SubtreeSize, Subtree Digest
- Annotations apply to individual hashes.
  - Examples: Size of all application data under the referenced object
- These are all optional. The draft does not make this terminology as clear cut as this slide does.

Some definitions

- Subtree Size: The bytes of application data at or under the current location
- Subtree Digest: cryptographic digest of that application data.
- Locators: Routing hints (full slide coming up)
- Leaf Size: Size of all application data immediately reachable from the current Hash Group (i.e. direct children)
- Leaf Digest: digest of that leaf data.

Locators and Name Constructors

- To construct an Interest given a hash, one needs to know the name prefix and/or routing hints if the data is available from other places.
- Locators can be used equally well by NDN and CCNx, though the protocol mechanisms differ.
- FLIC uses the concept of a Name Constructor to tell the consumer how to build an Interest from the hash, locators, or other names (e.g. based on the manifest name). FLIC defines 4 of them.

Hash Groups and Name Constructors

- Each Hash Group can have its own Name Constructor.
  - Example: Child manifests are available under /foo/control and application data is /foo/data. The manifest uses Hash Group 1 for child manifests and Hash group 2 for child data.
- There is a default NC.
- Locators and NCs are inherited, so one only needs to define them, say, in the root manifest.

Encryption

- FLIC defines its own encryption mechanism.
- The FLIC TLV/binary form is arranged so one could do in-place decryption and simply change a TLV type from “encrypted” to “plaintext”. Similarly, for encryption.
- We define two mechanism: pre-shared key and wrapped key. It is extensible and more mechanisms are welcome.

Preshared Key

- The manifest contains a key ID.
- The parties have exchanged keys, e.g. out-of-band or through a key exchange protocol.
- We use AEAD mechanism with AES-GCM or AES-CCM.

Wrapped Key

- This mode wraps an AES key, that could then be used by the AES mechanisms of the Preshared Key.
- It has a KeyId, a wrapped key, and a key locator for the KeyId.
- It uses RSA-OAEP.
- Future extensions are for RSA KemDem and Elliptic Curve.

Multiprotocol Use

- A manifest is an ABNF structure.
- It can be encoded in either NDN or CCNx using the native protocol TLVs.
- The FLIC draft provides CCNx and NDN encodings for the manifest and describes protocol-specific aspects of using FLIC.

Root vs Root + Top Manifest

- The draft describes using a Root manifest and also a Root + Top Manifest.
- The Root manifest must be named and/or discoverable. It has signatures, Locators, and Name Constructors.
- The Root + Top mechanism buids a manifest using the Top manifest. It then adds a Root over it. This technique allows swapping Root manifest, such as when Locators or NCs or signatures need to be updated, while all the rest of the manifest tree stays the same.

Still to be done

- The Python prototype is behind the draft.
- FLIC supports acyclic digraphs, not just trees. We need examples.
- The AEAD nonce needs to be spelled out more and be clear on RFC5288/RFC6655 usage vs NIST 800-38D terminology. We are not clear on the salt vs explicit nonce.
- For the RSA-OAEP section, we need to clarify the Nonce vs wrapped key + salt.
- We should consider using a KDF to ensure no repeated nonces between manifest nodes or whole manifest trees using the same keys.
- IANA considerations section.
- Security considerations section.
- We need to review the NDN section. It has not been touched in a long time and is likely out of date.