CCNx 1.0 Wire Format
ICNRG Interim Meeting, Boston
Marc Mosko, Palo Alto Research Center
Wire Format Goals

1. Unsigned part for network headers
2. Ability to sign without knowing signature length
3. No aliases of types
4. Independent Layer 3 operation
5. Fast to process
6. TLV types context-dependent
1. Unsigned part for network headers

- Some headers will change hop-to-hop
  - Interest lifetime
  - Priorities or class of service
  - HopLimit
- These should be grouped for efficiency and near the front of the packet, as only routers use them.
- They need to be outside security envelope
2. Ability to sign without knowing signature length

• Some signature algorithms have variable length signatures.

• The length of the signature should not change anything in the pieces being signed.
3. No aliases of types

- If variable length types, 0x01 ≠ 0x0001 ≠ 0x00000001, etc.

Boss: *Block that name /hacker/payroll*

Minion: *No problem.*

  ... create a signature for 0x08 0x06 ‘hacker’ ....

Hacker: *You are mine now!*

  ... create packet with 0xFD0008 0x06 ‘hacker’ ....
4. Independent Layer 3 operation

• Should be able to operate directly over L2
  – provide own loop detection
  – have total packet length
  – have own fragmentation
  – Own headers outside of upper-layer message
5. Fast to process

• Minimize branches
  – adversarial branching patterns up to 6x slower\(^1\)
• If variable length, know length before writing?
  – Construct packets backwards?
  – How do you know how far back to start?
• Works with kernel-bypass
  – Kernel buffers, where you need to start at “0”

6. TLV types context-dependent

- Vendor TLV containers should not depend on anyone else’s TLV types.
- Vendors should not need to register all their types, only the parent container type.
PARC Wire Format

- Fixed Header for L3 operation and fast access to Name, even with per-hop headers.
- Per-hop headers outside signature envelope.
- Signature payload own TLV at end, size only matters in Fixed Header ‘packetLength’.
- 2+2 TLV format, no aliases, fast.
- TLV types context-dependent.
## TLV PACKET

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Packet Type</td>
</tr>
<tr>
<td>PacketLength</td>
<td>Packet Length</td>
</tr>
<tr>
<td>HopLimit†</td>
<td>Hop Limit</td>
</tr>
<tr>
<td>reserved</td>
<td>Reserved Flag</td>
</tr>
<tr>
<td>Flags</td>
<td>Flags</td>
</tr>
<tr>
<td>HeaderLength</td>
<td>Header Length</td>
</tr>
<tr>
<td>~ Per-hop Header TLVs</td>
<td></td>
</tr>
<tr>
<td>~ CCNx Message TLV</td>
<td></td>
</tr>
<tr>
<td>~ Optional CCNx Validation Algorithm (VA) TLV</td>
<td></td>
</tr>
<tr>
<td>~ Optional CCNx Validation Payload (VP) TLV (required VA)</td>
<td></td>
</tr>
</tbody>
</table>

### PacketLength
- 0 to end of packet

### HeaderLength
- 0 to end of per-hop headers

### CCNx Message
- Interest / ContentObject / (other)

### ValidationAlgorithm
- CRC32C / Checksums / HMAC-SHA256 / VMAC-128 / RSA-SHA256 / EC-SECP-256K1 / EC-SECP-384R1

### ValidationPayload
- (validation payload)

### Validation Range
- (CCNx Message plus VA)

### Self-certified Name
- SHA-256(CCNx Message, VA, VP)

† HopLimit only applies to a PacketType Interest.
INTEREST

|   1   |   2   |   3   |
|-------+-------+-------|
| 0 1 2| 3 4 5| 6 7 8|
| 9 0 1| 2 3 4| 5 6 7|
| 8 9 0| 1 2 3| 4 5 6|
| 7 8 9| 0 1 2| 3 4 5|

| Version | Interest | PacketLength |
|---------+----------+-------------|
|         +----------+-------------|
| HopLimit | reserved | Flags       |
|         +----------+-------------|

~ Optional Header TLVs

| T_INTEREST | Interest Length |
|------------+-----------------|

~ Mandatory Name TLV

~ Optional Metadata TLV

~ Optional Payload TLV

~ Optional CCNx Validation Algorithm TLV

~ Optional CCNx Validation Payload TLV (required ValidationAlg)

Headers = Lifetime
Metadata = KeyIdRestriction / ContentObjectHashRestriction
INTEREST RETURN

~ Optional Header TLVs

~ Mandatory Name TLV

~ Optional Metadata TLV

~ Optional Payload TLV

~ Optional CCNx Validation Algorithm TLV

~ Optional CCNx Validation Payload TLV (required ValidationAlg)
CONTENT OBJECT

---

**Headers** = RecommendedCacheTime
**Metadata** = ExpiryTime / PayloadType
# Initial list of supported algorithms

ValidationType = CRC32C / RFC793 / HMAC-SHA256 / VMAC-128 / RSA-SHA256 / EC-SECP-256K1 / EC-SECP-384R1

CRC32C = (empty)
RFC793 = (empty)
HMAC-SHA261 = KeyId [SignatureTime]
RSA-SHA256 = KeyId [KeyLocator][SignatureTime]
EC-* = KeyId [KeyLocator][SignatureTime]
KeyLocator = PublicKey / Certificate / KeyName
PublicKey = (DER encoded public key)
Certificate = (DER encoded X.509 certificate)
KeyName = Link
Link = Name [KeyIdRestriction] [ContentObjectHashRestriction]
The validation payload depends on the verification type in the previous TLV.

For a CRC32, it’s the 32-bytes of CRC.
For an HMAC-SHA256, it is the 32-byte SHA256 output.
For RSA-SHA256, it’s the RSA signature.
Etc.