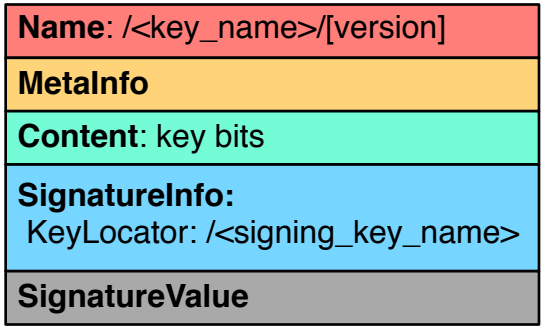
Multiple Signature

v0.3

Certificate

- A data packet binds key name to key bits
  - signed by certificate issuer

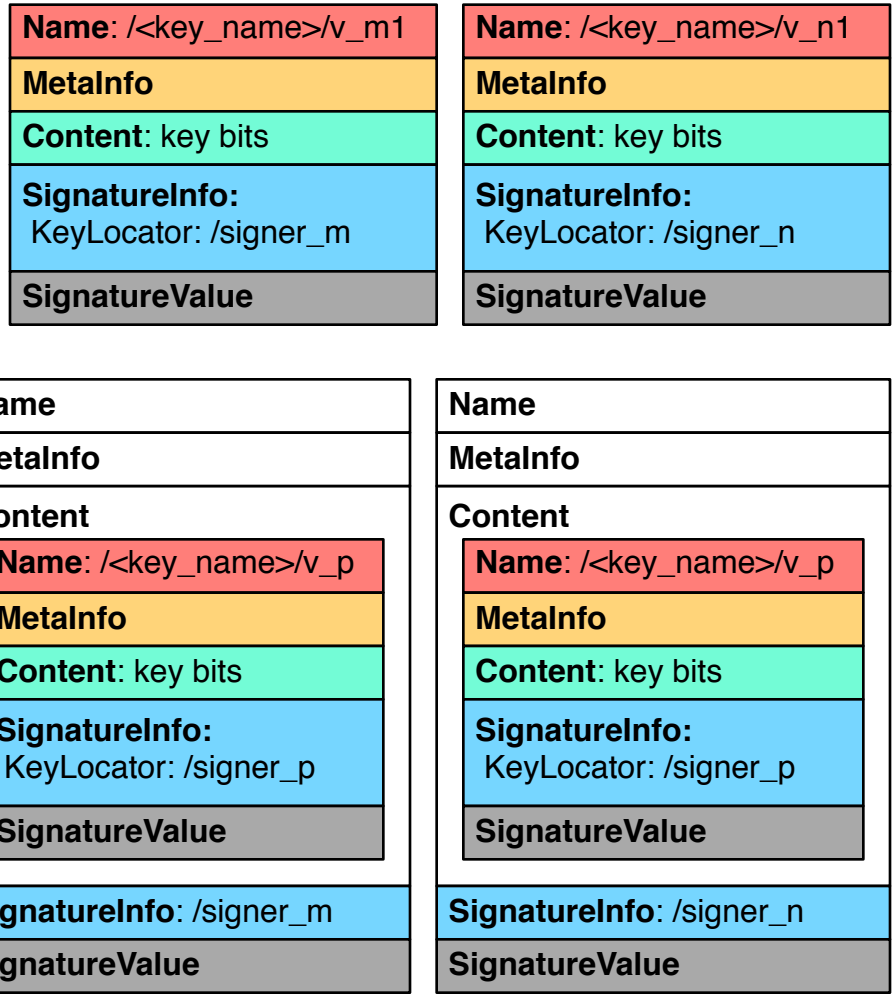


- Naming convention
  - /<key-name>/[version]
  - version
    - certificate issuer may replace its own key periodically
    - every time a new signing key is created, re-sign the binding between key name and key bits, leading to a certificate with new **version**

- Previously, we assume that there is only one issuer who can certify the binding between key name and key bits
  - version number is consistent from the issuer's perspective

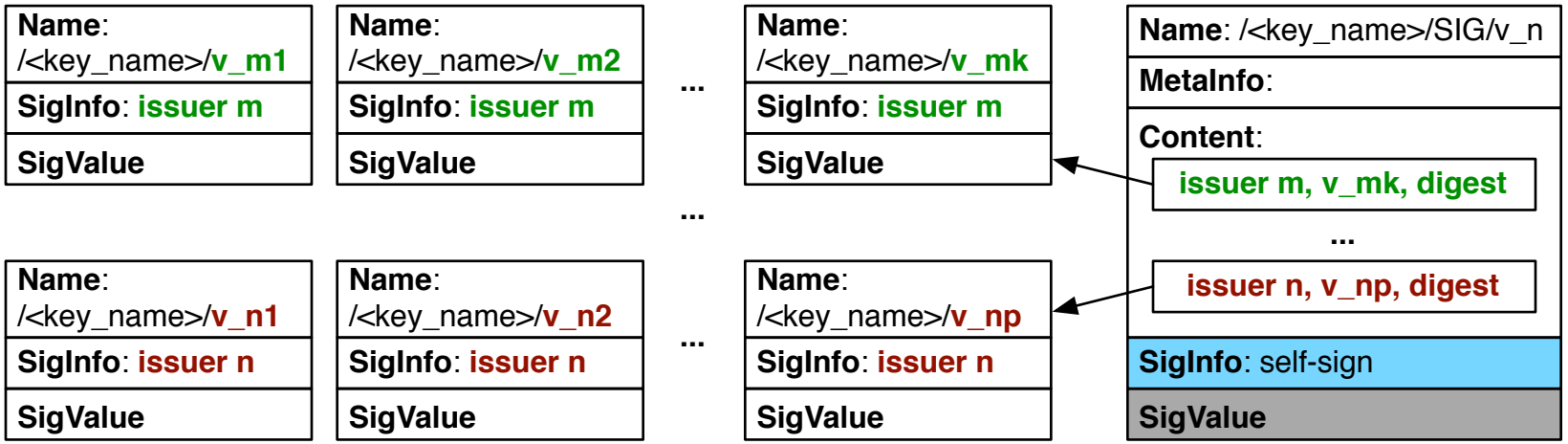
Multiple Signature

- Signature on the same (key name, key bits) binding?
  - how to maintain the version? or as long as version is consistent for each signer
  - v_m1 < v_m2 < v_m3 < …
  - v_n1 < v_n2 < v_n3 < …
- Signature on the same data packet?
  - encapsulation
  - who determine the inner version, sigInfo, and sigVal?
  - how to name the outer packet?
  - how to interpret such an encapsulation?



Signature Bundle

- Latest information about the certificates issued by all the issuers



Operations

- Certificate issuers
  - prepare certificates with single signature
  - make sure the certificate name is unique
    - through a issuer specific version
  - make certificates available for key owner to collect
- Key owners
  - collect certificates with single signatures
  - prepare signature bundles
  - make signature bundles available for verifier to retrieve
- Verifiers
  - follow the KeyLocator in Data packet to retrieve certificate with single signature
  - (optional) when multiple signature is needed, retrieve signature bundles

Issue Certificates

- Name
  - /<key_name>/[**issuerId**]/[**version**]
  - issuerId: the first n-bytes of hash of issuer name
    - if /ndn/ucla/cs/alexissues a cert for /ndn/ucla/cs/yingdi, the issuer id is hash(/ndn/ucla/cs/alex)[0:n-1]
    - when only a few issuers
      - hash could be simple (e.g., cityhash)
      - n could be small (e.g., 4)
  - version:
    - issuer specific
    - monotonically increase

- SignatureInfo: specified by issuer
  - e.g., ValiditidyPeriod, …

Certificate Collection

- Offline channel
  - when there are few certificates to collect at low frequency
- Cert collecting protocol
  - PGP-style Key Servers
    - issuer uploads its issued certificates to a key server (e.g., a repo)
    - key owners lookup certificates at the key server
  - Sync
    - treat all certificates for a single key as a data set to sync
  - NDNS record
    - an issuer puts its issued certificates as its own NDNS records for a key owner to collect
    - key owner knows its potential key
    - a NDNS record encapsulates one issued certificate

Serve Signature Bundle

- Prepare Signature bundle
  - format
    - name: /<key_name>/**SIG**/[**version**]
    - content:
      - a list of 3-tuples (issuer_name, latest_version, digest)
    - signed by key owner
  - when discover a new certificate
    - a new issuer or a existing issuer with new cert version
    - update tuple list & bundle version
  - when tuple list exceed MTU, segment bundle

- Key owner
  - serve certificates with single signature
  - serve signature bundle with the latest version

Certificate Retrieval

- Verifier extracts key name from KeyLocator of a data packet
- If verifier cannot determine the issuer name
  - send an interest /<key_name>
- If verifier can determine the issuer name
  - e.g. derived according to trust schema
  - send an interest: /<key_name>/[issuer_id]
- If verifier need multiple signatures
  - send an interest in parallel: /<key_name>/SIG
    - with MustBeFresh to get the latest version
  - on receiving signature bundle, a verifier learns all the issuers' name (and latest cert version), send specified interest for cert