Integrity of In-situ OAM Data Fields

draft-brockners-ippm-ioam-data-integrity-01

IETF 110 - IPPM WG, March 8th, 2021

Motivation

- The current [I-D.ietf-ippm-ioam-data] assumes that IOAM is deployed in specific network domains, where an operator has means to select, monitor, and control the access to all the networking devices, making the domain a trusted network. As such, IOAM tracing data is carried in the packets in clear and there are no protections against any node or middlebox tampering with the data.
- Recent discussions following the IETF last call on [I-D.ietf-ippm-ioam-data] revealed that there might be uses of IOAM where integrity protection of IOAM data fields is at least desirable, knowing that IOAM data fields integrity protection would incur extra effort in the data path of a device processing IOAM data fields.

Summary of threats considered

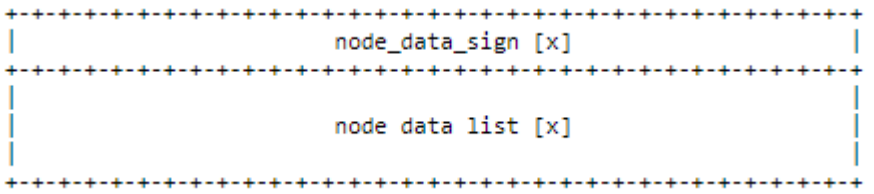
- In scope of the document
  - Modification: IOAM Data Fields
  - Modification: IOAM Option-Type Headers
  - Injection: IOAM Data Fields
  - Injection: IOAM Option-Type Headers
  - Replay (incl. “old” header with “new” data packet)
- Out of scope
  - Management and Exporting
  - Delay

Methods of providing integrity to IOAM data fields: Scope, Approach, Setup

- Scope:
  - Methods use Trace Option-Type data as example, but apply equally well to other Option-Types like E2E or DEX
- Approach:
  - Base idea is to add a new “signed node-data hash field” which is inserted by each node along with the node data to prove the integrity of the node data inserted
- Setup:
  - IOAM encap/decap/transit nodes
  - “Validator” node (typically the decap node) which verifies the integrity of the IOAM data fields.
  - Network management entity/controller which handles key distribution to the network nodes and serves as a receiver for validation results provided by the Validator.

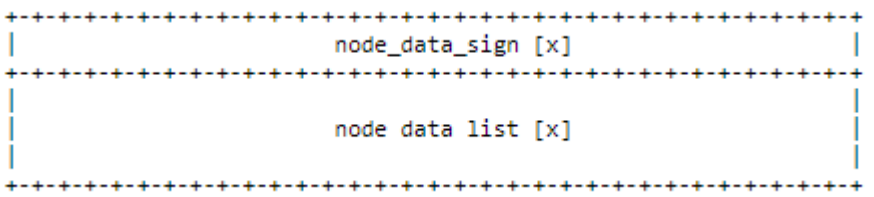
Methods of providing integrity to IOAM data fields: Overview

- Method 1: Using asymmetric keys for signing node trace
- Method 2: Using symmetric keys for signing node trace data
- Method 3: Space optimized symmetric key based signing of trace data
- Method 4: Dynamic symmetric key based signing of trace data
- Method 5: Leverage IP Authentication Header



Method 1:
Using asymmetric keys for signing node trace

1. Each IOAM capable node creates a key pair and shares the public key with the Validator and the controller/network management system responsible for using the IOAM trace information in the network domain.
2. Each `node data list [x]` field is extended with an additional "signed node-data" field: `node_data_sign[x]`. `Node_data_sign` includes a signature using the private key of the node over a hash of `node data list[x]` and a `node_data_sign[x-1]`.
3. IOAM encapsulating node will add a seed in its `node data list` that is used in its `node_data_sign`: The first IOAM node inserting the IOAM trace data will add `node_data_sign` over a "seed" || [hash of `node data` of first node].
4. The Validator uses the public key of each node to validate the signed node data elements in the same way the node trace signatures were created.

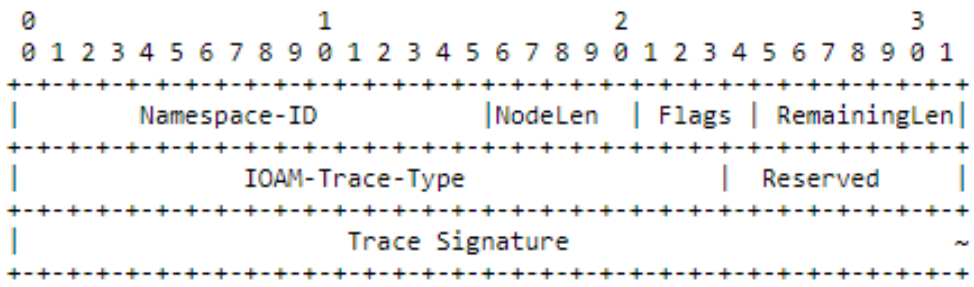


Method 2:
Using symmetric keys for signing node trace data

- Similar procedure to Method 1, using a Message Authentication Code (MAC) algorithm for the node signature.
- This involves distributing a secret key to the individual IOAM nodes and the Validator.
- Steps 1. to 4. of Method 1 apply in a similar way, i.e., each `node data list [x]` field is extended with an additional "signed node-data“ field: `node_data_sign[x]`

Method 3:
Space optimized symmetric key based signing of trace data

- Method uses symmetric keys with a **single** fixed size signature field (instead of per-hop signature field as with methods 1 and 2).
- Approach: Enhance the Option-Type header to carry the signature field. Example for Trace Option-Type:



Method 3 (cont'd):
Space optimized symmetric key based signing of trace data

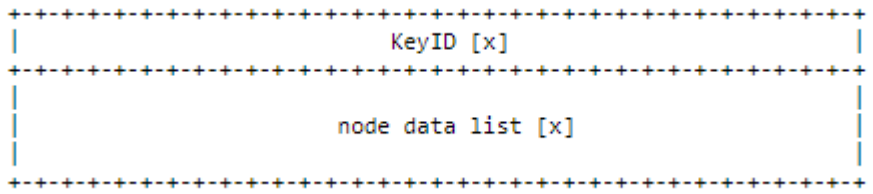
1. The first node creates a seed and sign/HMAC over the hash of its `node_data_list[x]`, the seed and its symmetric key. The seed can be included as a field in first node data or the seed can be the trailer to the trace option. The resulting HMAC/signature is included in the Trace Signature field.

2. Subsequent nodes will update the Trace Signature field by creating a signature/HMAC of data where the data is [Trace Signature ||hash of `node_data_list[x]`] with its symmetric key.

3. The Validator will iteratively recreate the Trace Signature over the node data trace fields collected and matches the Trace Signature field to validate the trace data integrity.

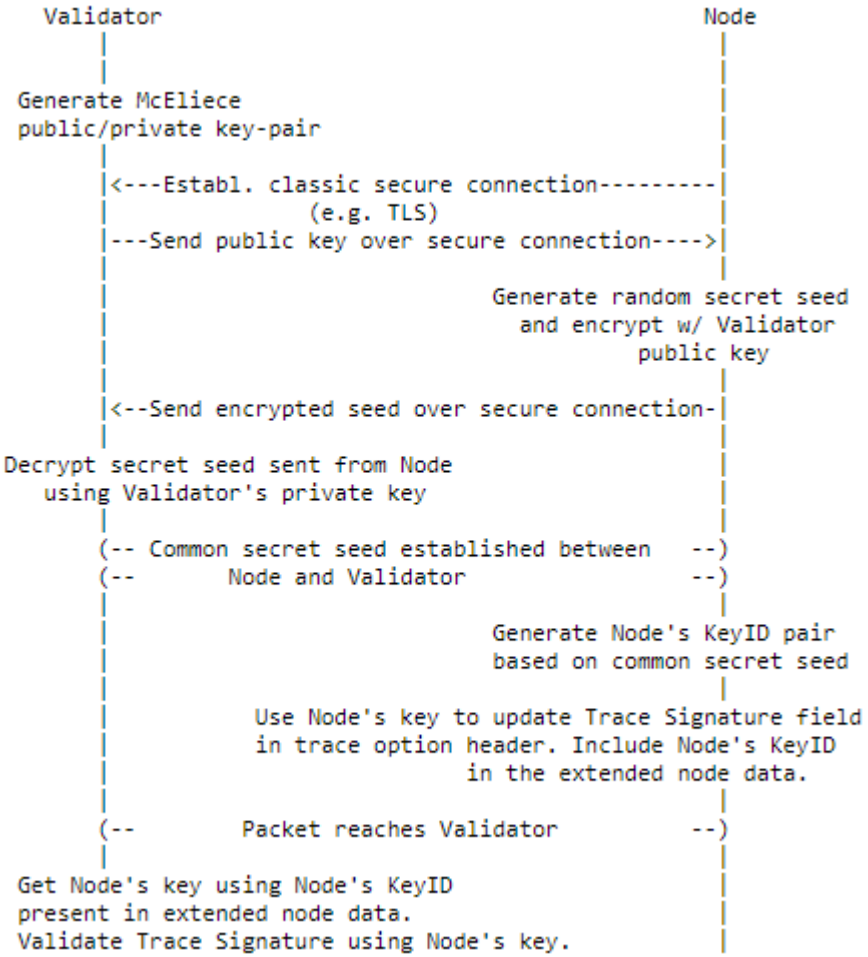
Method 4:
Dynamic symmetric key based signing of trace data

- Evolution of Method 3, leveraging Post-quantum Secure Pre-shared key distribution for deriving a dynamic symmetric key for every packet or a set of packets.
- Leverage local key generation service for Key/KeyID pairs for the participating Node and Validator.
  - This common key generator uses ratcheting cryptography to generate the next secret while forgetting about the previous one.
- A unique ID is paired with each secret generated.
  - Two implementations of the common key generator will generate the exact same key and associated ID when given the same seed secret as input parameter.
  - The common key generator can be queried for the next key or for a specific key ID.



Method 4:
Dynamic symmetric key based signing of trace data

1. Each node will establish a common secret seed establishment using the McEliece-Cryptosystem with the Validator.
2. Each node will then use the seed to generate a symmetric key per packet and use it in updating the `Trace Signature` field in the IOAM Trace-Option header over its node data hash. The node data is extended to include the KeyID of the dynamic key generated.
3. The Validator will validate the Trace Signature by deducing the key for each node using the KeyID.



Method 5:
Leverage IP Authentication Header

- The IP Authentication Header (AH) (RFC 4302) is used to provide connectionless integrity and data origin authentication for IP datagrams and to provide protection against replays.
- The AH provides authentication for as much of the IP header as possible, by classifying and including the immutable fields in the integrity calculation.
- To protect the IOAM data in the IP header, the AH may be employed in transport mode by setting up an IP AH Security Association (SA) with supported integrity algorithms between IOAM encapsulating nodes, IOAM decapsulating nodes and IOAM transit nodes.

Method 5 (cont'd): Applicability

- Method 5 is only applicable to IOAM Option-Types which immutable enroute, i.e.,
  - E2E Option-Type
  - Direct Export (DEX) Option-Type (Transit nodes which are to directly export IOAM data need to be configured with the negotiated shared secrets as well).
- Method 5 does not apply to
  - IOAM Pre-allocated Trace Option-Type
  - IOAM Incremental Trace Option-Type
  - IOAM Proof of Transit Option-Type

Discussion & Next Steps

- Discussion
  - Which method(s) does the WG prefer?
- Next Steps
  - WG adoption?