Delay/Disruption Tolerant Networking
Internet-Draft
Intended status: Standards Track
Expires: 16 September 2026

B. Dowling
King’s College London
B. Hale
X. Tian
Naval Postgraduate School
B. Wimalasiri
King’s College London

15 March 2026

Securing BPSec Against Arbitrary Packet Dropping
draft-tian-dtn-sbam-03

Abstract

In this document we describe Secure Bundle Protocol Audit Mechanism (SBAM), an authentication protocol designed to provide cryptographic auditing services for the Bundle Security protocol.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://bwimad.github.io/draft-xxx-str-bpsec/draft-tian-dtn-sbam.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-tian-dtn-sbam/.

Discussion of this document takes place on the Delay/Disruption Tolerant Networking Working Group mailing list (mailto:dtn@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/dtn/. Subscribe at https://www.ietf.org/mailman/listinfo/dtn/.

Source for this draft and an issue tracker can be found at https://github.com/bwimad/draft-xxx-str-bpsec.

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 https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction

This document defines additional security features for the Bundle Protocol Security (BPSec) [RFC9172] and is intended in use for Delay Tolerant Networking (DTN) environments using BPSec to provide security guarantees, Secure Bundle Audit Mechanism (SBAM) is intended to provide additional security guarantees for BPSec communication between a bundle source acting as origin security source, any intermediate security acceptors, and a destinaton security acceptor also acting as the bundle destination.

The BPSec specification [RFC9172] defines BPSec as "an end-to-end security service that operates in all of the environments where the BP operates" and claims to provide "integrity and confidentiality services for BP bundles". In particular, BPSec enables partial processing of bundles, where an intermediate node acting as a security acceptor can process and remove security services. As a result, it is possible for an intermediate malicious node to simply drop blocks along with any associated security operations attached to them.

SBAM provides in-band cryptographic integrity guarantees between a bundle source and its destination while remaining consistent with the operational requirements of BPSec, which permit intermediate nodes to process and discard security blocks added by the bundle source. At the same time, SBAM enables the destination node to detect adversarial modifications to security operations added to the bundle by the bundle source and to verify whether security service blocks added by the bundle source were maliciously dropped, processed, or modified during transit, while retaining compatibility with existing BPSec deployments.

1.1. Scope

This document defines a new security service for the Bundle Protocol Security (BPSec) [RFC9172] and is intended in use for Delay Tolerant Networking (DTN) environments using BPSec to provide security guarantees. Specifically, the Secure Bundle Audit Mechanism (SBAM) enables the cryptographic detection of unauthorized modifications to security operations added by the bundle source, which also acts as the security source for a destination node that accepts the bundle payload. Explicity, The end-to-end security guarantee provided by

SBAM is limited to security operations inserted by a bundle source acting as the security source for a destination node that accepts the bundle payload. Any security operations added to a bundle by security sources other than the bundle source are outside the scope of SBAM. In particular, security operations added by security sources along the bundle path between the bundle source and the destination, following bundle creation and the addition of SBAM operations, are legitimate and independent and are not covered by SBAM.

1.2. Notation

This section defines terminology that either is unique to the BPSec or SBAM and is necessary for understanding the concepts defined in this specification.

* Bundle Destination: the Bundle Protocol Agent (BPA) that receives a bundle and delivers the payload of the bundle to an Application Agent. Also, an endpoint comprising the node(s) at which the bundle is to be delivered. The bundle destination acts as the security acceptor for every security target in every security block in every bundle it receives.

* Bundle Protocol Agent: a node component that offers the Bundle Protocol services and executes its procedures.

* Bundle Source: the BPA that originates a bundle. Also, any node ID of the node of which the BPA is a component.

* Cipher Suite: a set of one or more algorithms providing integrity and/or confidentiality services. Cipher suites may define user parameters (e.g., secret keys to use), but they do not provide values for those parameters.

* Destination Node: a security acceptor BPA that is the bundle destination and processes the bundle payload.

* Forwarder: any BPA that transmits a bundle in DTN. Also, any node ID of the node of which the BPA that sent the bundle on its most recent hop is a component.

* Intermediate Node: a security acceptor BPA that is not the bundle destination.

* Intermediate Receiver, Waypoint, or Next Hop: any BPA that receives a bundle from a forwarder that is not the bundle destination. Also, any node ID of the node of which the BPA is a component.

* Path: the ordered sequence of nodes through which a bundle passes on its way from source to destination. The path is not necessarily known in advance by the bundle or any BPAs in DTN.

* Security Acceptor: a BPA that processes and dispositions one or more security blocks in a bundle. Security acceptors act as the endpoint of a security service represented in a security block. They remove the security blocks they act upon as part of processing and disposition. Also, any node ID of the node of which the BPA is a component.

* Security Block: a BPSec extension block in a bundle.

* Security Context: the set of assumptions, algorithms, configurations, and policies used to implement security services.

* Security Operation: the application of a given security service to a security target, denoted as OP(security service, security target). For example, OP(bcb-confidentiality, payload). Every security operation in a bundle MUST be unique, meaning that a given security service can only be applied to a security target once in a bundle. A security operation is implemented by a security block.

* Security Service: a process that gives some protection to a security target. For example, the BPSec specification defines security services for plaintext integrity (bib-integrity) and authenticated plaintext confidentiality with additional authenticated data (bcb-confidentiality). This SBAM specification defines security services for cryptographic auditing of security services added by the bundle source to the bundle destination.

* Security Source: a BPA that adds a security block to a bundle. Also, any node ID of the node of which the BPA is a component.

* Security Target: the block within a bundle that receives a security service as part of a security operation.

* Security Verifier: a BPA that verifies the data integrity of one or more security blocks in a bundle. Unlike security acceptors, security verifiers do not act as the endpoint of a security service, and they do not remove verified security blocks. Also, any node ID of the node of which the BPA is a component.

* Source Node: A BPA that creates an initial bundle.

* *Audit Pair*: A logical pairing consisting of a Manifest Block and its corresponding BIB, created by the bundle source acting as the initial security source and verified only by the destination node acting as the final security acceptor. The underlying Manifest Block records identifying data for each security operation added to the bundle by the bundle source.

* *Report Pair*: A logical pairing consisting of a Manifest Block and its corresponding BIB, created by an intermediate node that processed and discarded source-added blocks. The underlying Manifest Block duplicates identifying data for each bundle source寰殿dded security operation that is processed and discarded by an intermediate security acceptor.

1.3. Motivation and Problem Statement

DTN recognizes an attacker with complete network access, affording them read/write access to bundles traversing the network. Eavesdropping, modification, topological, and injection attacks are all described in [RFC9172], Section 8.2. Therein, these "on-path attackers" can be unprivileged, legitimate, or privileged nodes depending on their access to cryptographic material: unprivileged nodes only have access to publicly shared information, legitimate nodes have additional access to keys provisioned for itself, and privileged nodes have further access to keys (privately) provisioned for others. There are currently no guarantees against privileged attacks.

In an effort to distinguish malice by intermediate nodes, these classes can be further abstracted into honest security acceptors and dishonest forwarders. Honest forwarders are privileged nodes that faithfully execute the role of a BPA as described in [RFC9171], Section 3. Dishonest forwarders are unprivileged nodes that attempt to violate the integrity or confidentiality of blocks it processes (e.g. by dropping or modifying blocks). Under its default security context [RFC9173], BPSec currently provides no cryptographic auditing mechanism that enables a destination node to detect adversarial modifications to security services added to a bundle by the bundle source.

SBAM addresses this security gap by providing a mechanism that allows a bundle source, acting as the initial security source, to create a verifiable record of all security operations added to the bundle at origin, which is intended to be verified only by the final bundle destination. At the same time, SBAM preserves the default behavior of BPSec, allowing intermediate nodes to process and discard security operations added by the bundle source, provided they attach a verifiable record that duplicates the identifying data for each discarded security operation, which will subsequently be verified by the bundle destination.

2. Design Decisions

In this section we describe the design decisions of BPSec [RFC9172], and describe how these are impacted through the use of SBAM.

2.1. Block-Level Granularity

SBAM design does not impact the block-level granularity of BPSec. SBAM provides a verifiable audit trail between source and destination nodes, for all security blocks added by the source node, while also allowing intermediaries to process and discard source-added blocks.

2.2. Mixed Security Policy

SBAM design does not impact the mixed security policy of BPSec. SBAM design provides an additional layer of security between the source and destination nodes, by providing a mechanism for verifying that all security blocks added by the source node have either been processed by an authorized intermediary, or received by the destination node. This functionality does not interfere with the ability of additional security sources (that are not the bundle source) to create/modify/process security blocks within the bundle.

2.3. User-Defined Security Contexts

SBAM design does not impact the ability to implement user-defined security contexts within BPSec. Users may select from registered security contexts and customize those contexts through security context parameters.

2.4. Deterministic Processing

SBAM design preserves and adheres to the deterministic processing requirements described in [RFC9172].

2.5. COSE-Context Considerations

In conjunction with a proper PKI mechanism, SBAM may be used in the COSE-Context [draft-ietf-dtn-bpsec-cose] to provide further authentication enhancements to auditing. Specifically, through the use of digital signature algorithms rather than message authentication codes as described herein, SBAM in the COSE-context adds source authentication as well as authentication of intermediate nodes.

2.6. Scope Flag

The Integrity Security Context BIB_HMAC-SHA2 includes Integrity Scope Flags as a parameter set (see 3.2 and 3.3.3 in RFC9173). The value of the Integrity Scope Flag describes what information is used to construct the Integrity Protected Plain Text (IPPT) for a BIB. The existing Integrity Scope Flags in bit 2 and bit 3 refer to an excessive amount of information (block type code, block number, block processing control flags). Since we explicitly only use the block number in our calculations, this scope flag is redundant and we choose to remove it.

2.7. Security Blocks

In this section we describe the different Security Blocks used in BPSec and SBAM. In particular, we note that BPSec introduced two types of security blocks: the Block Integrity Block (BIB) and the Block Confidentiality Block (BCB) providing integrity and confidentiality and integrity, respectively.

In SBAM we also introduce the audit-pair logical block and the report-pair logical block, which (when combined) enable security acceptors to verify only honest intermediate security acceptors have processed and discarded BIB or BCBs added to a bundle at origin.

2.8. Block Definitions

The BPSec specification defines two types of security blocks: the Block Integrity Block (BIB) and the Block Confidentiality Block (BCB). The SBAM specification defines two additional types of logical blocks; the audit-pair and report-pair operations.

TODO: Check references are correct

* The BIB is used to ensure the integrity of its plaintext security target(s). The integrity information in the BIB MAY be verified by any node along the bundle path from the BIB security source to the bundle destination. Waypoints add or remove BIBs from bundles

in accordance with their security policy. BIBs are never used for integrity protection of the ciphertext provided by a BCB. Because security policy at BPSec nodes may differ regarding integrity verification, BIBs do not guarantee hop-by-hop authentication, as discussed in Section 1.1 (https://www.rfc-editor.org/rfc/rfc9172.html#sup_sec_svc).

* The BCB indicates that the security target or targets have been encrypted at the BCB security source in order to protect their content while in transit. As a matter of security policy, the BCB is decrypted by security acceptor nodes in the network, up to and including the bundle destination. BCBs additionally provide integrity-protection mechanisms for the ciphertext they generate.

3. SecureBPSec Audit Mechanism Protocol Overview

The core guarantee provided by SBAM is a guarantee that, after correctly verifying audit-pair and report-pair security operations, the destination node is assured that either

* all security blocks added by the bundle source have arrived without an unprivileged node dropping or modifying them; or

* an honest intermediary has processed and discarded security block/s added by the bundle source.

Thus, for any bundle, its source node also acting as initial security source, will generate security blocks for their destination node exactly as specified in BPSec [RFC9172]. Additionally, the source node will create a logical audit-pair block, which provides a cryptographically-authenticated record of all security services it provided for the bundle, as well as all necessary uniquely identifying information for each security operation, such as its key and block identifiers.

SBAM further specifies that any honest intermediary node that processes a security block created by the bundle source also provides a cryptographic proof that it was authorized to perform this operation. This is achieved by replacing the processed and discarded security operation with a report-pair logical block that duplicates and authenticates, to the destination node (the final security acceptor), the uniquely identifying information associated with the discarded security service contained in the security block. The SBAM report-pair therefore provides a cryptographically authenticated digest of the uniquely identifying information of the security block it processes, such as key and block identifiers. This allows the relevant identifying information of a bundle source寰殿dded security operation to remain verifiable by the final security acceptor even after the original security block has been processed and discarded.

Upon receiving the bundle, the destination node first verifies the audit-pair to validate its authenticity. It then verifies each report-pair to confirm that it was added by an honest intermediary node. The destination node subsequently collates the identifying information contained in the audit-pair and compares it with the identifying information collated from the report-pair blocks. Successful verification at each stage enables the destination node to confirm that no unprivileged node modified or removed any bundle source寰殿dded security operation during transit between the source and destination nodes.

3.1. Unique Key Identifiers

The Bundle Protocol Security (BPSec) and its defined security contexts, as described in RFC9172 [RFC9172] and RFC9173 [RFC9173] respectively, rely on the assumption that local security policies will inform Bundle Protocol Agents (BPAs) of the appropriate cryptographic keys to use for each security context. This decentralized, policy-driven approach allows flexibility but introduces ambiguity when these policies are not uniformly enforced or clearly defined across participating nodes. In the absence of standardized key selection mechanisms, there is a risk that different BPAs may select conflicting keys for the same security context or inadvertently reuse keys across incompatible contexts. Such ambiguity can lead to key collisions, where multiple security contexts reference the same key identifier or cryptographic material unintentionally, undermining the security operations BPSec is intended to enforce. To mitigate this ambiguity, our proposed SBAM mechanism introduces a key-id as an explicit security context parameter, enabling BPAs to uniquely identify the correct cryptographic key for each context.

Within the SBAM design, three independent use cases for key-id are identified. When combined, these use cases enable the establishment of a reciprocal trust relationship between the bundle source acting as the initial security source, privileged intermediary nodes, and the bundle destination acting as the final security acceptor. The three distinct but interconnected key-id use cases are as follows:

* key-id for BPSec security services : This identifier uniquely identifies the key used by the bundle source to provide a BPSec confidentiality or integrity service (BCB and BIB, respectively) when the bundle is created at origin. Each BPSec security service MUST have a corresponding unique key-id to facilitate SBAM integration.

* key-id for auditing : This identifier uniquely identifies the key used by the bundle source to authenticate the audit-pair logical block to the bundle destination. The audit-pair contains the uniquely identifying information for each security operation added to the bundle at origin, including the corresponding security service key-id values.

* key-id for reporting: This identifier uniquely identifies the key used by a privileged intermediary node to authenticate the report-pair logical block to the bundle destination. The report-pair duplicates the uniquely identifying information for each bundle source窶殿dded security operation (including the corresponding security service key-id values) that the intermediary processes and discards.

The use of these distinct key-id roles enables SBAM to bind the integrity of each bundle source窶殿dded BPSec security operation (via its associated security service key-id) to the key-id used to authenticate the audit-pair. Upon successful verification, the destination node can confirm that the integrity of the BPSec security operations added at the bundle origin has been preserved.

Independently, privileged intermediary nodes bind the report-pair service key-id to the uniquely identifying information of each corresponding security service key-id associated with a bundle source窶殿dded security block that they process and discard. This enables the destination node to verify that any modification to bundle source窶殿dded security operations was performed by a legitimate intermediary node.

The management of these key-id values, including how trust in them is established, maintained, and escrowed, is determined at the policy level.

Key identifiers are always represented as a CBOR unsigned integer. The CBOR encoding of values is as defined by the security context specification. Key identifiers MUST be unique across all security contexts and distinctly identify a cryptographic key used for a given security operation defined in its security context.

3.2. Bundle Protocol Manifest Block

The Bundle Protocol Manifest Block introduced in [sipos-dtn-manifest-block] defines a new structured block type for BPv7 bundles. A primary purpose of the Manifest Block is to provide a structured and auditable mechanism for maintaining a record of bundle security components between the bundle source, acting also as the security source, and the intended destination node. This mechanism allows honest intermediary nodes to act as legitimate security acceptors, enabling them to process, and discard security blocks added by the source node while preserving an auditable record of those security-related components within the proposed manifest structure. Manifest blocks enable the enumeration and identification of elements within a bundle in a consistent and machine-processable manner. While manifest blocks are not defined as security-specific mechanisms in itself, they provide a structured representation of bundle content that can be used by such security mechanisms as SBAM.

By listing covered components and associated metadata, manifest blocks create an environment in which integrity protection and authentication operations can be applied in a systematic way. In this sense, manifest blocks do not directly perform security functions, but it enables and supports such functions by supplying a well-defined container for describing and referencing BPSec bundle elements.

Below is a diagram of the proposed Manifest Block structure as defined in [sipos-dtn-manifest-block].

```
$$metadata-item //= (

0: int16 (manifest-reason)

2: embed-eid-structure

3: dtn-time)

$$blockdata-item //= (

1: block-id
block-id = [

; From the IANA Bundle Block Types registry

block-type: uint,

block-num: uint,]

2: block-control-flags

5: btsd-len
btsd-len = uint

6: [+ btsd-hash]

btsd-hash = (
; From the IANA COSE Algorithms registry
alg: tstr / int,
value: bstr
)

-1: bpsec-targets (for BPSec security blocks)
bpsec-targets = [+ uint ]

-2: int16 bpsec-security-context (for BPSec security blocks)
)

...
```

SBAM leverages and extends the proposed Manifest Block structure to provide a verifiable audit trail between the source node and the destination node. Manifest blocks enable SBAM functionality in two ways:

(1) they define an object type for the audit-pair operation between the bundle source and the final security-acceptor destination node; and

(2) they define an object type for the report-pair operation, which supports intermediary processing of source寰堵enerated security blocks while preserving the associated original audit information.

Together, these two object types establish a verifiable audit trail between the bundle source and its destination node. This audit trail preserves the BPSec-defined behavior that permits intermediary nodes to process and discard security blocks as required, while also providing a mechanism to detect any unauthorized modification to the security operations of a bundle enforced by its source.

3.3. Audit Creation

The audit creation process is described in detail in the following sections.

At the time of bundle creation, the source node SHALL generate a verifiable audit-pair logical block that covers all security operations added by the source node. Structurally, an audit-pair consists of a manifest block that records all security operations added to a bundle by its originator, and a BIB that authenticates the manifest contents.

The audit-pair SHALL contain all relevant identifying information for each relevant security block (represented by independent manifest block-data-map), which SHALL include at minimum the identity of the security block block-id, information about its security context including a unique key-id, a list of target block identifiers security-targets to which the security block operation applies, and the payload (encoded in btsd-hash) of the bundle. This audit-pair SHALL then be cryptographically authenticated by a BIB generated by the source node which computes a cryptographic MAC over the audit-pair using a unique audit-pair operation key shared with the destination node. The key-id for the BIB authenticating the audit-pair SHALL be included in the BIB security context and SHALL be independent of the security context information contained in the associated manifest block.

The destination node SHALL discard any audit-pair (along with any source-generated payload intended for it) that is not cryptographically verified by a BIB generated by the source node. A further flag MAY be added to the audit-pair and its BIB to indicate that it is expected only to be verified by the intended bundle destination security acceptor.

The uniquely identifying information associated with the BIB that authenticates the audit-pair MUST NOT be included in the audit-pair record, as this would introduce a circular verification dependency.

See SBAM Integration with Manifest Block (Section 3.7) for more details on audit-pair design specifications.

3.4. Reporting

The reporting process is described in detail in the following sections.

Any security accepting intermediary that processes an SBAM bundle security operation added by the bundle source SHALL replace the processed operation with a report-pair operation. Structurally, a report-pair consists of a manifest block, and a BIB that authenticates the manifest contents.

The report-pair SHALL contain all relevant identifying information for each security block processed and discarded by the intermediary along the bundle path. Such report-pair SHALL include at minimum the identity of the security block (block-id) processed, a duplicate record of its security context (including its unique security service key-id) and a duplicate record of the security targets to which the security block operation applies. This report-pair SHALL then be cryptographically authenticated by a BIB generated over the manifest which computes a cryptographic MAC over the manifest block-data-map using a unique report-pair operation key the processing intermediary shares with the destination node. The key-id for the BIB authenticating the report-pair SHALL be included in the BIB security context and SHALL be independent of the security context information contained in the associated manifest block.

A further flag MAY be added to the report-pair to indicate that it is expected only to be verified by the intended destination security acceptor.

The uniquely identifying information associated with the BIB that authenticates the report-pair MUST NOT be included in the report-pair record, as this would introduce a circular verification dependency.

See SBAM Integration with Manifest Block (Section 3.7) for more details on the report-pair design specifications.

3.5. Verification

The verification process is described in detail in the following sections.

When the destination node receives an SBAM bundle, it SHALL verify that the audit-pair and report-pair(s) it received can be cryptographically authenticated before accepting the bundle as valid. An SBAM bundle SHALL be considered valid by accepting destination node only if the following conditions are met:

1. The audit-pair is cryptographically verified by the attached BIB generated by the source node. Failure indicates tampering or corruption of the bundle or associated block-type-specific-data.

2. Each report-pair generated by an intermediary along the bundle path is cryptographically verified by its associated BIB. This enables the destination node to validate that the corresponding discarded bundle source-added security operation, which the report-pair replaced, was processed by an authorized intermediary and that the original security configuration of that operation remains cryptographically verifiable.

3. The block-type-specific-data of the audit-pair manifest block matches a concatenation of blockdata-item(s) for each report-pair manifest block.

3.6. Blocks Excluded by SBAM

SBAM participants should exclude blocks that necessarily change throughout a bundle’s life cycle from auditing. Extension blocks such as Hop-Count or Previous Node which change values SHOULD be excluded from SBAM protection to avoid unnecessary processing and overhead.

3.7. SBAM Integration with Manifest Block

Instead of defining a separate extension block, the SBAM can be integrated within the proposed Manifest Block structure. This approach leverages the existing manifest framework and would involve minor modifications or extensions to the Manifest Block fields currently defined in [sipos-dtn-manifest-block] to accommodate SBAM audit data.

When a manifest block is used as part of an audit-pair (see Audit Creation (Section 3.3)), the _reason_ item of the meta-data-map SHALL be set to security-sourcing. The resulting manifest-block SHALL then act as the security-target of a further BIB block which authenticates its content. This manifest block and its associated BIB block are generated by the initial bundle source and together form the logical unit referred to as an audit-pair. A further flag MAY be added to the audit-pair to indicate that it is expected only to be verified by the intended destination security acceptor. The BIB SHALL be

generated by the original source node and SHALL contain a cryptographic MAC over its associated manifest block using a unique audit-pair operation key shared with the destination node. Furthermore, an additional item that identifies the cryptographic key used for the corresponding cryptographic operation, bpsec-key-id, SHALL be added to the audit-pair when used for SBAM auditing. TODO: maybe extend figure 1 to depict the BIB block that verifies the audit manifest block.

```
$$metadata-item //= (

0: (int16) security-sourcing

2: embed-eid-structure

3: dtn-time)

// blockdata-item per each source-generated security block

$$blockdata-item //= (

1: block-id

2: block-control-flags

5: btsd-len

6: [+ btsd-hash]

-1: bpsec-targets

-2: bpsec-security-context

*-3: bpsec-key-id*
)

...

; TODO: specify EDN encoding details
```

Figure 1: Manifest Block structure adapted to facilitate SBAM auditing.

When a manifest block is used as part of a report-pair, the _reason_ item of the meta-data-map SHALL be set to security-acceptance. This report-pair is generated every time an intermediary acceptor processes a bundle source-generated security block. The resulting manifest-block SHALL then act as the security-target of a further BIB

block which authenticates its content. This manifest block and its associated BIB block are generated by the SBAM-processing intermediary and together form the logical unit referred to as an report-pair. A further flag MAY be added to the report-pair to indicate that it is expected only to be verified by the intended destination security acceptor.

Furthermore, an additional item that identifies the cryptographic key used for the corresponding cryptographic operation, bpsec-key-id, SHALL be added to the report-pair when used for SBAM reporting. This enables an intermediary security acceptor that processes a source-generated security block to append a verifiable record of the processed security-block-specific-data for use by the intended destination security acceptor. The btsd-len and btsd-hash fields in the manifest block do not serve a functional purpose within a report-pair and as such MAY be excluded without any functional impact.

This hop-by-hop generated report-pair SHALL subsequently be used by the intended destination security acceptor to verify the integrity of the received bundle.

```
$$metadata-item //= (

0: (int16) security-acceptance

2: embed-eid-structure

3: dtn-time)

// blockdata-item per each intermediary-processed source-generated security block

$$blockdata-item //= (

1: block-id

2: block-control-flags

-1: bpsec-targets

-2: bpsec-security-context

*-3: bpsec-key-id*
...
; TODO: specify EDN encoding details
```

Figure 2: Manifest Block structure adapted to facilitate SBAM reporting.

4. Processing Rules

* *source-node*: Adds audit-pair consisting of a manifest block along with its verifying BIB after other security blocks and before payload.

* *intermediate-node*: Processes BIB/BCB, adds report-pairconsisting of a manifest block along with its verifying BIB.

* *destination-node*: Validates all block-data-items within audit-pair and all report-pair(s), and checks whether the block-type-specific-data of the audit-pair manifest matches a concatenation of all block-type-specific-data for each report-pair manifest, prior to accepting payload. If any validation fails, the bundle MUST be discarded.

5. Security Considerations

5.1. Trivial Block Removal

SBAM allows for the detection of unauthorized deletion of source-added BIB/BCB. This requires that the intended recipient performing the SBAM verifications described in Section 3.5 to avoid a trivial attack by a malicious intermediary of simply removing all security blocks.

5.2. Insider Attack

SBAM protected bundles are still vulnerable to *privileged insider* attacks unless asymmetric crypto is introduced. Malicious nodes with privileged access to keys associated with protected blocks may be able to modify SBAM block values undetected (e.g. forge or overwrite report-container block-data-item report).

6. IANA Considerations

New block types:

* audit manifest block 寰TBD (see Bundle Protocol Manifest Block (Section 3.2) and [sipos-dtn-manifest-block])

* report-container manifest block TBD (see Bundle Protocol Manifest Block (Section 3.2) and [sipos-dtn-manifest-block])

7. References

7.1. Normative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/rfc/rfc2104>.

[RFC2119] Bradner, S. O., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC9171] Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol Version 7", RFC 9171, DOI 10.17487/RFC9171, January 2022, <https://www.rfc-editor.org/info/rfc9171>.

[RFC9172] Birrane, E. and K. Fall, "Bundle Protocol Security (BPSEC)", RFC 9172, DOI 10.17487/RFC9172, January 2022, <https://www.rfc-editor.org/info/rfc9172>.

[RFC9173] Birrane, E., "Bundle Protocol Security (BPSEC) Default Security Contexts", RFC 9173, DOI 10.17487/RFC9173, January 2022, <https://www.rfc-editor.org/info/rfc9173>.

7.2. Informative References

[CryptoRocket] "Cryptography is Rocket Science", n.d., <https://doi.org/10.62056/a39qudhdj>.

[draft-ietf-dtn-bpsec-cose] Sipos, B., "Bundle Protocol Security (BPSec) COSE Context", 3 June 2025, <https://datatracker.ietf.org/doc/draft-ietf-dtn-bpsec-cose/>.

[sipos-dtn-manifest-block] Sipos, B., "Bundle Protocol (BP) Manifest Block", 29 December 2025, <https://datatracker.ietf.org/doc/html/draft-sipos-dtn-manifest-block-00>.

Authors' Addresses

Benjamin Dowling
King's College London
Email: benjamin.dowling@kcl.ac.uk

Britta Hale
Naval Postgraduate School
Email: britta.hale@nps.edu

Xisen Tian
Naval Postgraduate School
Email: xisen.tian1@nps.edu

Bhagya Wimalasiri
King’s College London
Email: bhagya.wimalasiri@kcl.ac.uk