Delay/Disruption Tolerant Networking
Internet-Draft
Intended status: Informational
Expires: 9 August 2026

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

5 February 2026

Securing BPSec Against Arbitrary Packet Dropping
draft-tian-dtn-sbam-02

Abstract

In this document we describe Strong 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, SBAM is intended to provide additional security guarantees for BPSec communication between a security source (typically a bundle source), a security acceptor, and a security destination (typically 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 nodes to simply drop blocks (and associated security extension blocks). StrongBPSec Audit Mechanism (SBAM) provides in-band integrity guarantees by cryptographic auditing via ledger blocks, mitigating the risk of undetected message deletion in BPSec. Specifically, SBAM addresses a critical limitation of BPSec by enabling destination nodes to verify whether security service blocks added by the bundle source were dropped, processed, or modified during transit, while retaining compatibility with existing BPSec deployments.

1.1. 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.

* *BAB*: Bundle Audit Block a ledger block authenticated by the source node.

* *BRB*: Bundle Report Block a signed/verifiable block produced by an intermediate node that processed and discarded source-added blocks.

1.2. 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 intermmediate 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). This is the gap we address with SBAM: BPSec under the default security context [RFC9173] has no cryptographic auditing mechanism for detecting modifications to a bundle between the SN and DN. With SBAM, all security acceptors (including the intended destination) are obliged to record their modifications

2. Design Decisions

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

TODO: Use RFC 9172 draft as starting point, discuss how SBAM changes these, or our design does not effect.

2.1. Block-Level Granularity

2.2. Multiple Security Sources

2.3. Mixed Security Policy

2.4. User-Defined Security Contexts

2.5. Deterministic Processing

2.6. 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.

2.7. Unique Key Identifiers

The Bundle Protocol Security (BPSec) and its defined security contexts, as described in RFC9172 and 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_identifier as an explicit security context parameter, enabling BPAs to uniquely identify the correct cryptographic key for each context.

2.8. 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 describe 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.

2.9. 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 Bundle Audit Block (BAB) and the Block Report Block (BRB), which (when combined) enable security targets to verify only honest security intermediate nodes have processed missing BIB or BCBs.

2.10. 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 security blocks: the Bundle Audit Block (BAB) and the Block Report Block (BRB).

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. StrongBPSec Audit Mechanism Protocol Overview

The core guarantee provided by SBAM is a guarantee that, after correctly verifying the Bundle Audit Block, the destination node is assured that either

* all security blocks added by the Source Node have arrived without an unprivileged node dropping or modifying security block; or

* an honest intermediary has processed a security block.

Thus, for any bundle, the source node will generate all security blocks for their destination node exactly as in BPSec. Additionally, it will provide a Bundle Audit Block, which provides a cryptographically-authenticated digest of all security services it provided for the bundle, as well as all necessary uniquely identifying information, such as the key and block identifiers. This digest is added as the security result to the Bundle Audit Block.

Any honest intermediary node that processes a security block from the source bundle will also provide a cryptographic proof that they were privileged to perform this operation (demonstrated by providing a cryptographic signature over the identifying information for the security service contained in the security block). This signature is contained within the Block Report Block, which provides a cryptographically-authenticated digest of all uniquely identifying information of the security block it just processed, such as key and block identifiers.

Finally, when the destination receives the bundle, it will collate all identifying information contained within security blocks (generated by the source node) with identifying information contained within each BRB. Verifying this information against the cryptographic digest contained within the Bundle Audit Block enables the destination node to verify that no unprivileged node modified or dropped any security block between the source node and the destination node.

3.1. Bundle Audit Blocks (BAB)

This section describes the procedure used to compute a Message Authentication Code (MAC) tag for a bundle containing one or more security blocks added by the bundle SN.

Each SN MUST attach a BAB immediately before the Payload Block. This BAB contains metadata describing all security blocks added by the SN and a MAC computed using a key shared with the DN.

3.1.1. Inputs

* *plaintext* A binary string constructed as the concatenation of the payload and metadata elements from each security block added by the SN. Formally:

  plaintext = payload || { block_no || key_id || security_targets } for each security block added by the SN

  where: - payload: The application data block. - block_no: The identifier of the block being protected. - key_id: The identifier for the cryptographic key used as specified by the security context - security_targets: A list of target block numbers to which the security operation applies.

* *key* The symmetric key (e.g., a pre-shared key or key-wrapped ephemeral key) used to compute the MAC. This key is identified by the key_id and MUST be established in accordance with the security context shared between communicating nodes.

3.1.2. Output

* *MAC tag* A cryptographic tag that provides data origin authentication and integrity verification. The MAC is calculated as follows:

  MAC = HMAC(key, plaintext)

  where: - HMAC is the keyed-hash message authentication code function as defined in [RFC2104].

3.1.3. Usage

The MAC tag generated by this procedure (alongside the plaintext) is attached to the BAB security block as the security_result field and MUST be verified by the DN. Failure to validate the tag indicates tampering or corruption of the bundle or associated metadata.

TODO: Describe how to reconstruct the BAB payload from the security blocks (e.g. BIB, BCB, BRB) present in the rest of the bundle.

3.1.4. Block Specific Data

TODO: describe the exact sub-fields for the security block. Broadly it follows the following structure:

* Number of Security Blocks (Integer)

* Key identifier (BAB key shared between the SN and DN)

* For each security block represented in the BAB:

  - Block number (of the original Security Block)

  - Key identifier (of the original Security Block)

  - Security Targets (of the original Security Block)

  - Block Type Code (of the original Security Block)

  - Security Parameters (of the original Security Block)

Note that the (Block, Key, Target, Code, Parameters) field should be ordered by block number to ensure consistent ordering between the SN generating the MAC tag and the DN verifying the MAC tag.

3.2. Bundle Report Blocks (BRB)

This section defines how a Message Authentication Code (MAC) is generated by an IN when processing and discarding a security block from a bundle. The resulting tag is attached to a newly created Block Report Block (BRB). This enables the DN to verify the legitimacy of IN(s) that process and discarded SN-originated added security blocks.

An IN that processes and discards any SN-originated security block MUST add a BRB. Each BRB cryptographically authenticates metadata about the discarded block (e.g., key_id, block_id, security_targets) and is authenticated with a unique key (independant of the BAB key) shared with the DN.

3.2.1. Inputs

* *plaintext* A structured binary string composed of identifying information related to the discarded block:

  plaintext = block_no || key_id || security_targets

  where: - block_no: The unique identifier of the discarded security block. - key_id: The identifier of the key used by the SN for the original security block. - security_targets: A list of block numbers to which the discarded security block originally applied.

* *key* A symmetric key used for computing the MAC. This may be a pre-shared key or a key-wrapped ephemeral variant, as specified by the active security context.

3.2.2. Output

* *MAC tag* A cryptographic tag calculated over the plaintext:

  MAC = HMAC(key, plaintext)

  where: - HMAC is the keyed-hash message authentication code function defined in [RFC2104].

3.2.3. Usage

The resulting MAC tag, along with the original plaintext, is attached to the *Read Report Block (BRB)* as the security results field. This allows the destination node to validate that the discarded security block was processed by an authorized intermediate node, and that its original security configuration is cryptographically verifiable.

3.2.4. Block Specific Data

TODO: describe the exact sub-fields for the security block. Broadly it follows the following structure:

* Block number (of the original Security Block)
* Key identifier (of the original Security Block)
* Security Targets (of the original Security Block)
* Block Type Code (of the original Security Block)
* Security Parameters (of the original Security Block)
* MAC tag

3.3. 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.

3.4. Integration with BIB/BCB

Existing BIB and BCB behavior is preserved. BAB and BRBs are *in-band* and protect against message deletion or replacement without the need for out-of-band policies.

''' +==============================+ | Primary Bundle Block | +==============================+ | Version | | Bundle Processing Flags | | Destination EID | | Source EID | | Report-to EID | | Creation Timestamp | | Lifetime | | (Optional Fragment Offset) | | (Optional Total App Data) | +==============================+

+==============================+ | Block Report Block | +==============================+ | Block Type Code = xxxx | | Block Number = # | | Block Processing Flags | | EID References | | Security Source EID | | Security Parameters (e.g., | | hash algorithm, keys) | | Security Results (e.g., | | Read Receipt HMAC) | | Security Targets (block #s) | +==============================+

+==============================+ | Generic Extension Block | +==============================+ | Block Type Code = xxxx | | Block Number = # | | Block Processing Flags | | EID References (if any) | | Payload Length | | Payload Data | +==============================+

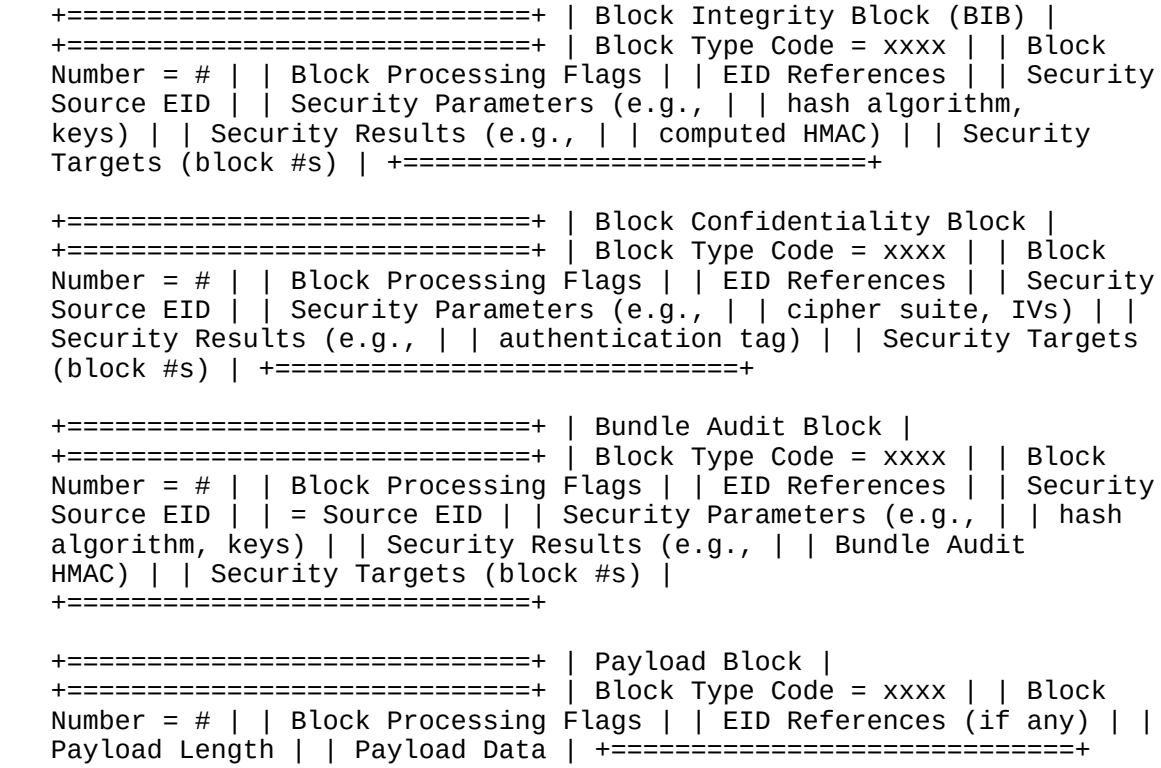

Figure 1. SBAM protected bundle ‘’’

4. Processing Rules

* *SN*: Adds BAB after security blocks and before Payload.

* *IN*: Processes BIB/BCB, replaces with BRB as needed.

* *DN*: Validates all BRBs and BAB prior to accepting payload. If any validation fails, bundle MUST be discarded.

5. Security Considerations

5.1. Trivial Block Removal

SBAM allows for the detection of unauthorized deletion of SN-added BIB/BCB. This requires that the intended recipient checks for a BAB and rejects bundles without one as to avoid a trivial attack by a malicious IN 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 undected (e.g. forge or overwrite BRB values).

6. IANA Considerations

New block types: - BAB TBD - BRB TBD

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>.

[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/>.

Appendix A. Abstract Security Block Representation

A.1. BAB

An example of the abstract security block structure based on Fig. 1 of the BAB is as follows. We assume the block numbers are sequential for the sake of illustration.

[4, 5, 7], / Blocks logged - BIB, BCB, Payload block number / 5, / Security Context ID - BAB-HMAC-SHA2 / 1, / Security Context Flags - Parameters Present / [2,[2, 1]], / Security Source - ipn:2.1 / [/ Security Parameters - 3 Parameters / [1, 6], / SHA Variant - HMAC 384/384 / [5, h'0xf000ba4'] / Key Identifier - Unique to BAB context /], [/ Security Results: 1 Result / [/ Target 1 Results / [1, h'deadbeefdeadbeefdeadbeefdeadbeef / MAC / deadbeefdeadbeefdeadbeefdeadbeef deadbeefdeadbeefdeadbeefdeadbeef']]]

A.2. BRB

An example of the abstract security block structure of the BRB is as follows.

[5], / Discarded Block - BCB block number / 5, / Security Context ID - BRB-HMAC-SHA2 / 1, / Security Context Flags - Parameters Present / [2,[2, 2]], / Security Source - ipn:2.2 / [/ Security Parameters - 3 Parameters / [1, 6], / SHA Variant - HMAC 384/384 / [5, h'0xcafebabe'] / Key Identifier - Unique to BRB context /], [/ Security Results: 1 Result / [/ Target 1 Results / [1, h'deadbeefdeadbeefdeadbeefdeadbeef / MAC / deadbeefdeadbeefdeadbeefdeadbeef deadbeefdeadbeefdeadbeefdeadbeef']]]

Authors' Addresses

Benjamin Dowling
Kings 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