Stream: Internet Engineering Task Force (IETF)
RFC: 9172
Category: Standards Track
Published: January 2022
ISSN: 2070-1721
Authors: E. Birrane, III *JHU/APL* K. McKeever *JHU/APL*

RFC 9172
Bundle Protocol Security (BPSec)

Abstract

This document defines a security protocol providing data integrity and confidentiality services for the Bundle Protocol (BP).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines security features for the Bundle Protocol (BP) [RFC9171] and is intended for use in Delay-Tolerant Networking (DTN) to provide security services between a security source and a security acceptor. When the security source is the bundle source and the security acceptor is the bundle destination, the security service provides end-to-end protection.

The Bundle Protocol specification [RFC9171] defines DTN as referring to "a network architecture providing communications in and/or through highly stressed environments" where "BP may be viewed as sitting at the application layer of some number of constituent networks, forming a store-carry-forward overlay network". The phrase "stressed environment" refers to multiple challenging conditions including intermittent connectivity, large and/or variable delays, asymmetric data rates, and high bit error rates.

It should be presumed that the BP will be deployed in an untrusted network, which poses the usual security challenges related to confidentiality and integrity. However, the stressed nature of the BP operating environment imposes unique conditions where usual transport security mechanisms may not be sufficient. For example, the store-carry-forward nature of the network may require protecting data at rest, preventing unauthorized consumption of critical resources such as storage space, and operating without regular contact with a centralized security oracle (such as a certificate authority).

An end-to-end security service that operates in all of the environments where the BP operates is needed.

1.1. Supported Security Services

BPSec provides integrity and confidentiality services for BP bundles, as defined in this section.

Integrity services ensure that changes to target data within a bundle can be discovered. Data changes may be caused by processing errors, environmental conditions, or intentional manipulation. In the context of BPSec, integrity services apply to plaintext in the bundle.

Confidentiality services ensure that target data is unintelligible to nodes in DTN, except for authorized nodes possessing special information. Generally, this means producing ciphertext from plaintext and generating authentication information for that ciphertext. In this context, confidentiality applies to the contents of target data and does not extend to hiding the fact that confidentiality exists in the bundle.

NOTE: Hop-by-hop authentication is NOT a supported security service in this specification, for two reasons:

1. The term "hop-by-hop" is ambiguous in a BP overlay, as nodes that are adjacent in the overlay may not be adjacent in physical connectivity. This condition is difficult or impossible to detect; therefore, hop-by-hop authentication is difficult or impossible to enforce.

2. Hop-by-hop authentication cannot be deployed in a network if adjacent nodes in the network have incompatible security capabilities.

1.2. Specification Scope

This document defines the security services provided by the BPSec. This includes the data specification for representing these services as BP extension blocks and the rules for adding, removing, and processing these blocks at various points during the bundle's traversal of a delay-tolerant network.

BPSec addresses only the security of data traveling over the DTN, not the underlying DTN itself. Furthermore, while the BPSec protocol can provide security-at-rest in a store-carry-forward network, it does not address threats that share computing resources with the DTN and/or BPSec software implementations. These threats may be malicious software or compromised libraries that intend to intercept data or recover cryptographic material. Here, it is the responsibility of the BPSec implementer to ensure that any cryptographic material, including shared secrets or private keys, is protected against access within both memory and storage devices.

Completely trusted networks are extremely uncommon. Among untrusted networks, different networking conditions and operational considerations require security mechanisms of varying strengths. Mandating a single security context, which is a set of assumptions, algorithms, configurations, and policies used to implement security services, may result in too much security for some networks and too little security in others. Default security contexts are defined in [RFC9173] to provide basic security services for interoperability testing and for operational use on the terrestrial Internet. It is expected that separate documents will define different security contexts for use in different networks.

This specification addresses neither the fitness of externally defined cryptographic methods nor the security of their implementation.

This specification does not address the implementation of security policies and does not provide a security policy for the BPSec. Similar to cipher suites, security policies are based on the nature and capabilities of individual networks and network operational concepts. This specification does provide policy considerations that can be taken into account when building a security policy.

With the exception of the Bundle Protocol, this specification does not address how to combine the BPSec security blocks with other protocols, other BP extension blocks, or other best practices to achieve security in any particular network implementation.

1.3. Related Documents

This document is best read and understood within the context of the following other DTN documents:

- "Delay-Tolerant Networking Architecture" [RFC4838] defines the architecture for DTN and identifies certain security assumptions made by existing Internet protocols that are not valid in DTN.

- "Bundle Protocol Version 7" [RFC9171] defines the format and processing of bundles, the extension block format used to represent BPSec security blocks, and the canonical block structure used by this specification.
- "Concise Binary Object Representation (CBOR)" [RFC8949] defines a data format that allows for small code size, fairly small message size, and extensibility without version negotiation. The block-type-specific data associated with BPSec security blocks is encoded in this data format.
- "Bundle Security Protocol Specification" [RFC6257] introduces the concept of using BP extension blocks for security services in DTN. BPSec is a continuation and refinement of this document.

1.4. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This section defines terminology that either is unique to the BPSec or 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 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.

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 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, notated 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, this specification defines security services for plaintext integrity (bib-integrity) and authenticated plaintext confidentiality with additional authenticated data (bcb-confidentiality).

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.

2. Design Decisions

The application of security services in DTN is a complex endeavor that must consider physical properties of the network (such as connectivity and propagation times), policies at each node, application security requirements, and current and future threat environments. This section identifies those desirable properties that guide design decisions for this specification and that are necessary for understanding the format and behavior of the BPSec protocol.

2.1. Block-Level Granularity

Security services within this specification must allow different blocks within a bundle to have different security services applied to them.

Blocks within a bundle represent different types of information. The primary block contains identification and routing information. The payload block carries application data. Extension blocks carry a variety of data that may augment or annotate the payload or that otherwise provide information necessary for the proper processing of a bundle along a path. Therefore, applying a single level and type of security across an entire bundle fails to recognize that blocks in a bundle represent different types of information with different security needs.

For example, a payload block might be encrypted to protect its contents and an extension block containing summary information related to the payload might be integrity signed but unencrypted to provide waypoints access to payload-related data without providing access to the payload.

2.2. Multiple Security Sources

A bundle can have multiple security blocks, and these blocks can have different security sources. BPSec implementations **MUST NOT** assume that all blocks in a bundle have the same security operations applied to them.

The Bundle Protocol allows extension blocks to be added to a bundle at any time during its existence in DTN. When a waypoint adds a new extension block to a bundle, that extension block **MAY** have security services applied to it by that waypoint. Similarly, a waypoint **MAY** add a security service to an existing block, consistent with its security policy.

When a waypoint adds a security service to the bundle, the waypoint is the security source for that service. The security block(s) that represent that service in the bundle may need to record this security source, as the bundle destination might need this information for processing.

For example, a bundle source may choose to apply an integrity service to its plaintext payload. Later a waypoint node, representing a gateway to another portion of the delay-tolerant network, may receive the bundle and choose to apply a confidentiality service. In this case, the integrity security source is the bundle source and the confidentiality security source is the waypoint node.

In cases where the security source and security acceptor are not the bundle source and bundle destination, respectively, it is possible that the bundle will reach the bundle destination prior to reaching a security acceptor. In cases where this may be a practical problem, it is recommended that solutions such as bundle encapsulation be used to ensure that a bundle be delivered to a security acceptor prior to being delivered to the bundle destination. Generally, if a bundle reaches a waypoint that has the appropriate configuration and policy to act as a security acceptor for a security service in the bundle, then the waypoint should act as that security acceptor.

2.3. Mixed Security Policy

The security policy enforced by nodes in the delay-tolerant network may differ.

Some waypoints will have security policies that require the waypoint to evaluate security services even if the waypoint is neither the bundle destination nor the final intended acceptor of the service. For example, a waypoint could choose to verify an integrity service even though the waypoint is not the bundle destination and the integrity service will be needed by other nodes along the bundle's path.

Some waypoints will determine, through policy, that they are the intended recipient of the security service and will terminate the security service in the bundle. For example, a gateway node could determine that, even though it is not the destination of the bundle, it should verify and remove a particular integrity service or attempt to decrypt a confidentiality service, before forwarding the bundle along its path.

Some waypoints could understand security blocks but refuse to process them unless they are the bundle destination.

2.4. User-Defined Security Contexts

A security context is the set of assumptions, algorithms, configurations, and policies used to implement security services. Different contexts may specify different algorithms, different polices, or different configuration values used in the implementation of their security services. BPSec provides a mechanism to define security contexts. Users may select from registered security contexts and customize those contexts through security context parameters.

For example, some users might prefer a SHA2 hash function for integrity, whereas other users might prefer a SHA3 hash function. Providing either separate security contexts or a single, parameterized security context allows users flexibility in applying the desired cipher suite, policy, and configuration when populating a security block.

2.5. Deterministic Processing

Whenever a node determines that it must process more than one security block in a received bundle (either because the policy at a waypoint states that it should process security blocks or because the node is the bundle destination), the order in which security blocks are processed must be deterministic. All nodes must impose this same deterministic processing order for all security blocks. This specification provides determinism in the application and evaluation of security services, even when doing so results in a loss of flexibility.

3. Security Blocks

3.1. Block Definitions

This specification defines two types of security block: the Block Integrity Block (BIB) and the Block Confidentiality Block (BCB).

- 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.
- 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.

3.2. Uniqueness

Security operations in a bundle **MUST** be unique; the same security service **MUST NOT** be applied to a security target more than once in a bundle. Since a security operation is represented by a security block, this means that multiple security blocks of the same type cannot share the same security targets. A new security block **MUST NOT** be added to a bundle if a preexisting security block of the same type is already defined for the security target of the new security block.

This uniqueness requirement ensures that there is no ambiguity related to the order in which security blocks are processed or how security policy can be specified to require certain security services be present in a bundle.

Using the notation OP(service, target), several examples illustrate this uniqueness requirement.

Signing the payload twice: The two operations OP(bib-integrity, payload) and OP(bib-integrity, payload) are redundant and **MUST NOT** both be present in the same bundle at the same time.

Signing different blocks: The two operations OP(bib-integrity, payload) and OP(bib-integrity, extension_block_1) are not redundant and both may be present in the same bundle at the same time. Similarly, the two operations OP(bib-integrity, extension_block_1) and OP(bib-integrity, extension_block_2) are also not redundant and may both be present in the bundle at the same time.

Different services on same block: The two operations OP(bib-integrity, payload) and OP(bcb-confidentiality, payload) are not inherently redundant and may both be present in the bundle at the same time, pursuant to other processing rules in this specification.

Different services from different block types: The notation OP(service, target) refers specifically to a security block, as the security block is the embodiment of a security service applied to a security target in a bundle. Were some Other Security Block (OSB) to be defined providing an integrity service, then the operations OP(bib-integrity, target) and OP(osb-integrity, target) **MAY** both be present in the same bundle if so allowed by the definition of the OSB, as discussed in Section 10.

NOTES:

- A security block may be removed from a bundle as part of security processing at a waypoint node with a new security block being added to the bundle by that node. In this case, conflicting security blocks never coexist in the bundle at the same time and the uniqueness requirement is not violated.
- A ciphertext integrity-protection mechanism (such as associated authenticated data) calculated by a cipher suite and transported in a BCB is considered part of the confidentiality service; therefore, it is unique from the plaintext integrity service provided by a BIB.

- The security blocks defined in this specification (BIB and BCB) are designed with the intention that the BPA adding these blocks is the authoritative source of the security service. If a BPA adds a BIB on a security target, then the BIB is expected to be the authoritative source of integrity for that security target. If a BPA adds a BCB to a security target, then the BCB is expected to be the authoritative source of confidentiality for that security target. More complex scenarios, such as having multiple nodes in a network sign the same security target, can be accommodated using the definition of custom security contexts (see Section 9) and/or the definition of OSBs (see Section 10).

3.3. Target Multiplicity

A single security block **MAY** represent multiple security operations as a way of reducing the overall number of security blocks present in a bundle. In these circumstances, reducing the number of security blocks in the bundle reduces the amount of redundant information in the bundle.

A set of security operations can be represented by a single security block when all of the following conditions are true.

- The security operations apply the same security service. For example, they are all integrity operations or all confidentiality operations.
- The security context parameters for the security operations are identical.
- The security source for the security operations is the same, meaning the set of operations are being added by the same node.
- No security operations have the same security target, as that would violate the need for security operations to be unique.
- None of the security operations conflict with security operations already present in the bundle.

When representing multiple security operations in a single security block, the information that is common across all operations is represented once in the security block; the information that is different (e.g., the security targets) is represented individually.

If a node processes any security operation in a security block, it is **RECOMMENDED** that it process all security operations in the security block. This allows security sources to assert that the set of security operations in a security block are expected to be processed by the same security acceptor. However, the determination of whether a node actually is a security acceptor or not is a matter of the policy of the node itself. In cases where a receiving node determines that it is the security acceptor of only a subset of the security operations in a security block, the node may choose to only process that subset of security operations.

3.4. Target Identification

A security target is a block in the bundle to which a security service applies. This target must be uniquely and unambiguously identifiable when processing a security block. The definition of the extension block header from [RFC9171] provides a "block number" field suitable for this purpose. Therefore, a security target in a security block **MUST** be represented as the block number of the target block.

3.5. Block Representation

Each security block uses the Canonical Bundle Block Format as defined in [RFC9171]. That is, each security block is comprised of the following elements:

- block type code
- block number
- block processing control flags
- cyclic redundancy check (CRC) type
- block-type-specific data
- CRC field (if present)

Security-specific information for a security block is captured in the block-type-specific data field.

3.6. Abstract Security Block

The structure of the security-specific portions of a security block is identical for both the BIB and BCB block types. Therefore, this section defines an Abstract Security Block (ASB) data structure and discusses its definition, its processing, and other constraints for using this structure. An ASB is never directly instantiated within a bundle, it is only a mechanism for discussing the common aspects of BIB and BCB security blocks.

The fields of the ASB **SHALL** be as follows, listed in the order in which they must appear. The encoding of these fields **MUST** be in accordance with the canonical forms provided in Section 4.

Security Targets:
: This field identifies the block(s) targeted by the security operation(s) represented by this security block. Each target block is represented by its unique block number. This field **SHALL** be represented by a Concise Binary Object Representation (CBOR) array of data items. Each target within this CBOR array **SHALL** be represented by a CBOR unsigned integer. This array **MUST** have at least one entry and each entry **MUST** represent the block number of a block that exists in the bundle. There **MUST NOT** be duplicate entries in this array. The order of elements in this list has no semantic meaning outside of the context of this block. Within the block, the ordering of targets must match the ordering of results associated with these targets.

Security Context Id:
: This field identifies the security context used to implement the security service represented by this block and applied to each security target. This field **SHALL** be represented by a CBOR unsigned integer. The values for this Id should come from the registry defined in Section 11.3.

Security Context Flags:
: This field identifies which optional fields are present in the security block. This field **SHALL** be represented as a CBOR unsigned integer whose contents shall be interpreted as a bit field. Each bit in this bit field indicates the presence (bit set to 1) or absence (bit set to 0) of optional data in the security block. The association of bits to security block data is defined as follows.

Bit 0	(the least-significant bit, 0x01): "Security context parameters present" flag.
Bit >0	Reserved

Implementations **MUST** set reserved bits to 0 when writing this field and **MUST** ignore the values of reserved bits when reading this field. For unreserved bits, a value of 1 indicates that the associated security block field **MUST** be included in the security block. A value of 0 indicates that the associated security block field **MUST NOT** be in the security block.

Security Source:
: This field identifies the BPA that inserted the security block in the bundle. Also, any node ID of the node of which the BPA is a component. This field **SHALL** be represented by a CBOR array in accordance with the rules in [RFC9171] for representing endpoint IDs (EIDs).

Security Context Parameters (Optional):
: This field captures one or more security context parameters that should be used when processing the security service described by this security block. This field **SHALL** be represented by a CBOR array. Each entry in this array is a single security context parameter. A single parameter **SHALL** also be represented as a CBOR array comprising a 2-tuple of the Id and value of the parameter, as follows.

Parameter Id: This field identifies which parameter is being specified. This field **SHALL** be represented as a CBOR unsigned integer. Parameter Ids are selected as described in Section 3.10.

Parameter Value: This field captures the value associated with this parameter. This field **SHALL** be represented by the applicable CBOR representation of the parameter, in accordance with Section 3.10.

The logical layout of the parameters array is illustrated in Figure 1.

```
     +-----------------+-----------------+     +-----------------+
     |  Parameter 1    |  Parameter 2    | ... |  Parameter N    |
     +------+----------+------+----------+     +------+----------+
     |  Id  |  Value   |  Id  |  Value   |     |  Id  |  Value   |
     +------+----------+------+----------+     +------+----------+
```

Figure 1: Security Context Parameters

Security Results:

This field captures the results of applying a security service to the security targets of the security block. This field **SHALL** be represented as a CBOR array of target results. Each entry in this array represents the set of security results for a specific security target. The target results **MUST** be ordered identically to the Security Targets field of the security block. This means that the first set of target results in this array corresponds to the first entry in the Security Targets field of the security block, and so on. There **MUST** be one entry in this array for each entry in the Security Targets field of the security block.

The set of security results for a target is also represented as a CBOR array of individual results. An individual result is represented as a CBOR array comprising a 2-tuple of a result Id and a result value, defined as follows.

Result Id: This field identifies which security result is being specified. Some security results capture the primary output of a cipher suite. Other security results contain additional annotative information from cipher suite processing. This field **SHALL** be represented as a CBOR unsigned integer. Security result Ids will be as specified in Section 3.10.

Result Value: This field captures the value associated with the result. This field **SHALL** be represented by the applicable CBOR representation of the result value, in accordance with Section 3.10.

The logical layout of the security results array is illustrated in Figure 2. In this figure, there are N security targets for this security block. The first security target contains M results and the Nth security target contains K results.

```
   +---------------------------+     +---------------------------+
   |          Target 1         |     |         Target N          |
   +----------+----+-----------+     +---------------------------+
   | Result 1 |    | Result M  | ... | Result 1 |    |  Result K |
   +----+-----+ .. +----+------+     +---+------+ .. +----+------+
   | Id |Value|    | Id |Value|      | Id |Value|    | Id | Value|
   +----+-----+    +----+------+     +----+-----+    +----+------+
```

Figure 2: Security Results

3.7. Block Integrity Block

A BIB is a BP extension block with the following characteristics.

- The block type code value is as specified in Section 11.1.
- The block-type-specific data field follows the structure of the ASB.
- A security target listed in the Security Targets field **MUST NOT** reference a security block defined in this specification (e.g., a BIB or a BCB).
- The security context **MUST** utilize an authentication mechanism or an error detection mechanism.

Notes:

- Designers **SHOULD** carefully consider the effect of setting flags that either discard the block or delete the bundle in the event that this block cannot be processed.
- Since OP(bib-integrity, target) is allowed only once in a bundle per target, it is **RECOMMENDED** that users wishing to support multiple integrity-protection mechanisms for the same target define a multi-result security context. Such a context could generate multiple security results for the same security target using different integrity-protection mechanisms or different configurations for the same integrity-protection mechanism.
- A BIB is used to verify the plaintext integrity of its security target. However, a single BIB **MAY** include security results for blocks other than its security target when doing so establishes a needed relationship between the BIB security target and other blocks in the bundle (such as the primary block).
- Security information **MAY** be checked at any hop on the way to the bundle destination that has access to the required keying information, in accordance with Section 3.9.

3.8. Block Confidentiality Block

A BCB is a BP extension block with the following characteristics.

- The block type code value is as specified in Section 11.1.
- The block processing control flags value can be set to whatever values are required by local policy with the following exceptions:
  - BCBs **MUST** have the "Block must be replicated in every fragment" flag set if one of the targets is the payload block. Having that BCB in each fragment indicates to a receiving node that the payload portion of each fragment represents ciphertext.
  - BCBs **MUST NOT** have the "Block must be removed from bundle if it can't be processed" flag set. Removing a BCB from a bundle without decrypting its security targets removes information from the bundle necessary for their later decryption.
- The block-type-specific data fields follow the structure of the ASB.
- A security target listed in the Security Targets field can reference the payload block, a non-security extension block, or a BIB. A BCB **MUST NOT** include another BCB as a security target.

A BCB **MUST NOT** target the primary block. A BCB **MUST NOT** target a BIB unless it shares a security target with that BIB.

- Any security context used by a BCB **MUST** utilize a confidentiality cipher that provides authenticated encryption with associated data (AEAD).
- Additional information created by a cipher suite (such as an authentication tag) can be placed either in a security result field or in the generated ciphertext. The determination of where to place this information is a function of the cipher suite and security context used.

The BCB modifies the contents of its security target(s). When a BCB is applied, the security target body data are encrypted "in-place". Following encryption, the security target block-type-specific data field contains ciphertext, not plaintext.

Notes:

- It is **RECOMMENDED** that designers carefully consider the effect of setting flags that delete the bundle in the event that this block cannot be processed.
- The BCB block processing control flags can be set independently from the processing control flags of the security target(s). The setting of such flags should be an implementation/policy decision for the encrypting node.

3.9. Block Interactions

The security block types defined in this specification are designed to be as independent as possible. However, there are some cases where security blocks may share a security target; this sharing creates processing dependencies.

If a BCB and a BIB share a security target, an undesirable condition occurs: a waypoint would be unable to validate the BIB because the shared security target has been encrypted by the BCB. To address this situation, the following processing rules **MUST** be followed:

- When adding a BCB to a bundle, if some (or all) of the security targets of the BCB match all of the security targets of an existing BIB, then the existing BIB **MUST** also be encrypted. This can be accomplished either by adding a new BCB that targets the existing BIB or by adding the BIB to the list of security targets for the BCB. Deciding which way to represent this situation is a matter of security policy.
- When adding a BCB to a bundle, if some (or all) of the security targets of the BCB match some (but not all) of the security targets of a BIB, then that BIB **MUST** be altered in the following way. Any security results in the BIB associated with the BCB security targets **MUST** be removed from the BIB and placed in a new BIB. This newly created BIB **MUST** then be encrypted. The encryption of the new BIB can be accomplished either by adding a new BCB that targets the new BIB or by adding the new BIB to the list of security targets for the BCB. Deciding which way to represent this situation is a matter of security policy.
- A BIB **MUST NOT** be added for a security target that is already the security target of a BCB as this would cause ambiguity in block processing order.
- A BIB integrity value **MUST NOT** be checked if the BIB is the security target of an existing BCB. In this case, the BIB data is encrypted.

- A BIB integrity value **MUST NOT** be checked if the security target associated with that value is also the security target of a BCB. In such a case, the security target data contains ciphertext as it has been encrypted.
- As mentioned in Section 3.7, a BIB **MUST NOT** have a BCB as its security target.

These restrictions on block interactions impose a necessary ordering when applying security operations within a bundle. Specifically, for a given security target, BIBs **MUST** be added before BCBs. This ordering **MUST** be preserved in cases where the current BPA is adding all of the security blocks for the bundle or where the BPA is a waypoint adding new security blocks to a bundle that already contains security blocks.

In cases where a security source wishes to calculate both a plaintext integrity-protection mechanism and encrypt a security target, a BCB with a security context that generates an integrity-protection mechanism as one or more additional security results **MUST** be used instead of adding both a BIB and then a BCB for the security target at the security source.

3.10. Parameter and Result Identification

Each security context **MUST** define its own context parameters and results. Each defined parameter and result is represented as the tuple of an identifier and a value. Identifiers are always represented as a CBOR unsigned integer. The CBOR encoding of values is as defined by the security context specification.

Identifiers **MUST** be unique for a given security context but do not need to be unique amongst all security contexts.

An example of a security context can be found in [RFC9173].

3.11. BPSec Block Examples

This section provides two examples of BPSec blocks applied to bundles. In the first example, a single node adds several security operations to a bundle. In the second example, a waypoint node received the bundle created in the first example and adds additional security operations. In both examples, the first column represents blocks within a bundle and the second column represents the block number for the block, using the terminology B1...Bn for the purpose of illustration.

3.11.1. Example 1: Constructing a Bundle with Security

In this example, a bundle has four non-security-related blocks: the primary block (B1), two extension blocks (B4, B5), and a payload block (B6). The bundle source wishes to provide an integrity signature of the plaintext associated with the primary block, the second extension block, and the payload. The bundle source also wishes to provide confidentiality for the first extension block. The resultant bundle is illustrated in Figure 3 and the security actions are described below.

```
                    Block in Bundle                 ID
+==========================================+====+
|              Primary Block                | B1 |
+------------------------------------------+----+
|                    BIB                    | B2 |
|   OP(bib-integrity, targets = B1, B5, B6)|    |
+------------------------------------------+----+
|                    BCB                    | B3 |
|    OP(bcb-confidentiality, target = B4)  |    |
+------------------------------------------+----+
|       Extension Block (encrypted)         | B4 |
+------------------------------------------+----+
|              Extension Block              | B5 |
+------------------------------------------+----+
|               Payload Block               | B6 |
+------------------------------------------+----+
```

Figure 3: Security at Bundle Creation

The following security actions were applied to this bundle at its time of creation.

- An integrity signature applied to the canonical form of the primary block (B1), the canonical form of the block-type-specific data field of the second extension block (B5), and the canonical form of the payload block (B6). This is accomplished by a single BIB (B2) with multiple targets. A single BIB is used in this case because all three targets share a security source, security context, and security context parameters. Had this not been the case, multiple BIBs could have been added instead.
- Confidentiality for the first extension block (B4). This is accomplished by a BCB (B3). Once applied, the block-type-specific data field of extension block B4 is encrypted. The BCB **MUST** hold an authentication tag for the ciphertext either in the ciphertext that now populates the first extension block or as a security result in the BCB itself, depending on which security context is used to form the BCB. A plaintext integrity signature may also exist as a security result in the BCB if one is provided by the selected confidentiality security context.

3.11.2. Example 2: Adding More Security at a New Node

Consider that the bundle as it is illustrated in Figure 3 is now received by a waypoint node that wishes to encrypt the second extension block and the bundle payload. The waypoint security policy is to allow existing BIBs for these blocks to persist, as they may be required as part of the security policy at the bundle destination.

The resultant bundle is illustrated in Figure 4 and the security actions are described below. Note that block IDs provided here are ordered solely for the purpose of this example and are not meant to impose an ordering for block creation. The ordering of blocks added to a bundle **MUST** always be in compliance with [RFC9171].

```
                 Block in Bundle                ID
+==========================================+====+
|              Primary Block               | B1 |
+------------------------------------------+----+
|                    BIB                   | B2 |
|      OP(bib-integrity, target = B1)      |    |
+------------------------------------------+----+
|                    BIB (encrypted)       | B7 |
|      OP(bib-integrity, targets = B5, B6) |    |
+------------------------------------------+----+
|                    BCB                   | B8 |
|OP(bcb-confidentiality,targets = B5,B6,B7)|    |
+------------------------------------------+----+
|                    BCB                   | B3 |
|    OP(bcb-confidentiality, target = B4)  |    |
+------------------------------------------+----+
|       Extension Block (encrypted)        | B4 |
+------------------------------------------+----+
|       Extension Block (encrypted)        | B5 |
+------------------------------------------+----+
|         Payload Block (encrypted)        | B6 |
+------------------------------------------+----+
```

Figure 4: Security at Bundle Forwarding

The following security actions were applied to this bundle prior to its forwarding from the waypoint node.

- Since the waypoint node wishes to encrypt the block-type-specific data field of blocks B5 and B6, it **MUST** also encrypt the block-type-specific data field of the BIBs providing plaintext integrity over those blocks. However, BIB B2 could not be encrypted in its entirety because it also held a signature for the primary block (B1). Therefore, a new BIB (B7) is created and security results associated with B5 and B6 are moved out of BIB B2 and into BIB B7.
- Now that there is no longer confusion about which plaintext integrity signatures must be encrypted, a BCB is added to the bundle with the security targets being the second extension block (B5) and the payload (B6) as well as the newly created BIB holding their plaintext integrity signatures (B7). A single new BCB is used in this case because all three targets share a security source, security context, and security context parameters. Had this not been the case, multiple BCBs could have been added instead.

4. Canonical Forms

Security services require consistency and determinism in how information is presented to cipher suites at security sources, verifiers, and acceptors. For example, integrity services require that the same target information (e.g., the same bits in the same order) is provided to the cipher suite when generating an original signature and when validating a signature. Canonicalization algorithms transcode the contents of a security target into a canonical form.

Canonical forms are used to generate input to a security context for security processing at a BP node. If the values of a security target are unchanged, then the canonical form of that target will be the same even if the encoding of those values for wire transmission is different.

BPSec operates on data fields within bundle blocks (e.g., the block-type-specific data field). In their canonical form, these fields **MUST** include their own CBOR encoding and **MUST NOT** include any other encapsulating CBOR encoding. For example, the canonical form of the block-type-specific data field is a CBOR byte string existing within the CBOR array containing the fields of the extension block. The entire CBOR byte string is considered the canonical block-type-specific data field. The CBOR array framing is not considered part of the field.

The canonical form of the primary block is as specified in [RFC9171] with the following constraint.

- CBOR values from the primary block **MUST** be canonicalized using the rules for Deterministically Encoded CBOR, as specified in [RFC8949].

All non-primary blocks share the same block structure and are canonicalized as specified in [RFC9171] with the following constraints.

- CBOR values from the non-primary block **MUST** be canonicalized using the rules for Deterministically Encoded CBOR, as specified in [RFC8949].
- Only the block-type-specific data field may be provided to a cipher suite for encryption as part of a confidentiality security service. Other fields within a non-primary block **MUST NOT** be encrypted or decrypted and **MUST NOT** be included in the canonical form used by the cipher suite for encryption and decryption. An integrity-protection mechanism **MAY** be applied to these other fields as supported by the security context. For example, these fields might be treated as associated authenticated data.
- Reserved and unassigned flags in the block processing control flags field **MUST** be set to 0 in a canonical form as it is not known if those flags will change in transit.

Security contexts **MAY** define their own canonicalization algorithms and require the use of those algorithms over the ones provided in this specification. In the event of conflicting canonicalization algorithms, algorithms defined in a security context take precedence over this specification when constructing canonical forms for that security context.

5. Security Processing

This section describes the security aspects of bundle processing.

5.1. Bundles Received from Other Nodes

Security blocks must be processed in a specific order when received by a BP node. The processing order is as follows.

- When BIBs and BCBs share a security target, BCBs **MUST** be evaluated first and BIBs second.

5.1.1. Receiving BCBs

If a received bundle contains a BCB, the receiving node **MUST** determine whether it is the security acceptor for any of the security operations in the BCB. If so, the node **MUST** process those operations and remove any operation-specific information from the BCB prior to delivering data to an application at the node or forwarding the bundle. If processing a security operation fails, the target **SHALL** be processed according to the security policy. A bundle status report indicating the failure **MAY** be generated. When all security operations for a BCB have been removed from the BCB, the BCB **MUST** be removed from the bundle.

If the receiving node is the destination of the bundle, the node **MUST** decrypt any BCBs remaining in the bundle. If the receiving node is not the destination of the bundle, the node **MUST** process the BCB if directed to do so as a matter of security policy.

If the security policy of a node specifies that a node should have applied confidentiality to a specific security target and no such BCB is present in the bundle, then the node **MUST** process this security target in accordance with the security policy. It is **RECOMMENDED** that the node remove the security target from the bundle because the confidentiality (and possibly the integrity) of the security target cannot be guaranteed. If the removed security target is the payload block, the bundle **MUST** be discarded.

If an encrypted payload block cannot be decrypted (i.e., the ciphertext cannot be authenticated), then the bundle **MUST** be discarded and processed no further. If an encrypted security target other than the payload block cannot be decrypted, then the associated security target and all security blocks associated with that target **MUST** be discarded and processed no further. In both cases, requested status reports (see [RFC9171]) **MAY** be generated to reflect bundle or block deletion.

When a BCB is decrypted, the recovered plaintext for each security target **MUST** replace the ciphertext in each of the security targets' block-type-specific data fields. If the plaintext is of a different size than the ciphertext, the framing of the CBOR byte string of this field must be updated to ensure this field remains a valid CBOR byte string. The length of the recovered plaintext is known by the decrypting security context.

If a BCB contains multiple security operations, each operation processed by the node **MUST** be treated as if the security operation has been represented by a single BCB with a single security operation for the purposes of report generation and policy processing.

5.1.2. Receiving BIBs

If a received bundle contains a BIB, the receiving node **MUST** determine whether it is the security acceptor for any of the security operations in the BIB. If so, the node **MUST** process those operations and remove any operation-specific information from the BIB prior to delivering data to an application at the node or forwarding the bundle. If processing a security operation fails, the target **SHALL** be processed according to the security policy. A bundle status report indicating the failure **MAY** be generated. When all security operations for a BIB have been removed from the BIB, the BIB **MUST** be removed from the bundle.

A BIB **MUST NOT** be processed if the security target of the BIB is also the security target of a BCB in the bundle. Given the order of operations mandated by this specification, when both a BIB and a BCB share a security target, it means that the security target must have been encrypted after it was integrity signed; therefore, the BIB cannot be verified until the security target has been decrypted by processing the BCB.

If the security policy of a node specifies that a node should have applied integrity to a specific security target and no such BIB is present in the bundle, then the node **MUST** process this security target in accordance with the security policy. It is **RECOMMENDED** that the node remove the security target from the bundle if the security target is not the payload or primary block. If the security target is the payload or primary block, the bundle **MAY** be discarded. This action can occur at any node that has the ability to verify an integrity signature, not just the bundle destination.

If a receiving node is not the security acceptor of a security operation in a BIB, it **MAY** attempt to verify the security operation anyway to prevent forwarding corrupt data. If the verification fails, the node **SHALL** process the security target in accordance with local security policy. If a payload integrity check fails at a waypoint, it is **RECOMMENDED** that it be processed in the same way as a failure of a payload integrity check at the bundle destination. If the check passes, the node **MUST NOT** remove the security operation from the BIB prior to forwarding.

If a BIB contains multiple security operations, each operation processed by the node **MUST** be treated as if the security operation has been represented by a single BIB with a single security operation for the purposes of report generation and policy processing.

5.2. Bundle Fragmentation and Reassembly

If it is necessary for a node to fragment a bundle payload, and security services have been applied to that bundle, the fragmentation rules described in [RFC9171] **MUST** be followed. As defined there and summarized here for completeness, only the payload block can be fragmented; security blocks, like all extension blocks, can never be fragmented.

Due to the complexity of payload-block fragmentation, including the possibility of fragmenting payload-block fragments, integrity and confidentiality operations are not to be applied to a bundle representing a fragment. Specifically, a BCB or BIB **MUST NOT** be added to a bundle if the "Bundle is a fragment" flag is set in the bundle processing control flags field.

Security processing in the presence of payload-block fragmentation may be handled by other mechanisms outside of the BPSec protocol or by applying BPSec blocks in coordination with an encapsulation mechanism. A node should apply any confidentiality protection prior to performing any fragmentation.

6. Key Management

There exists a myriad of ways to establish, communicate, and otherwise manage key information in DTN. Certain DTN deployments might follow established protocols for key management, whereas other DTN deployments might require new and novel approaches. BPSec assumes that key management is handled as a separate part of network management; this specification neither defines nor requires a specific strategy for key management.

7. Security Policy Considerations

When implementing BPSec, several policy decisions must be considered. This section describes key policies that affect the generation, forwarding, and receipt of bundles that are secured using this specification. No single set of policy decisions is envisioned to work for all secure DTN deployments.

- If a bundle is received that contains combinations of security operations that are disallowed by this specification, the BPA must determine how to handle the bundle: the bundle may be discarded, the block affected by the security operation may be discarded, or one security operation may be favored over another.
- BPAs in the network must understand what security operations they should apply to bundles. This decision may be based on the source of the bundle, the destination of the bundle, or some other information related to the bundle.
- If a waypoint has been configured to add a security operation to a bundle, and the received bundle already has the security operation applied, then the receiver must understand what to do. The receiver may discard the bundle, discard the security target and associated BPSec blocks, replace the security operation, or take some other action.
- It is **RECOMMENDED** that security operations be applied to every block in a bundle and that the default behavior of a BPA be to use the security services defined in this specification. Designers should only deviate from the use of security operations when the deviation can be justified -- such as when doing so causes downstream errors when processing blocks whose contents must be inspected or changed at one or more hops along the path.
- BCB security contexts can alter the size of extension blocks and the payload block. Security policy **SHOULD** consider how changes to the size of a block could negatively effect bundle processing (e.g., calculating storage needs and scheduling transmission times).
- Adding a BIB to a security target that has already been encrypted by a BCB is not allowed. If this condition is likely to be encountered, there are (at least) three possible policies that could handle this situation.

  1. At the time of encryption, a security context can be selected that computes a plaintext integrity-protection mechanism that is included as a security context result field.
  2. The encrypted block may be replicated as a new block with a new block number and may be given integrity protection.

3. An encapsulation scheme may be applied to encapsulate the security target (or the entire bundle) such that the encapsulating structure is, itself, no longer the security target of a BCB and may therefore be the security target of a BIB.

- Security policy **SHOULD** address whether cipher suites whose ciphertext is larger than the initial plaintext are permitted and, if so, for what types of blocks. Changing the size of a block may cause processing difficulties for networks that calculate block offsets into bundles or predict transmission times or storage availability as a function of bundle size. In other cases, changing the size of a payload as part of encryption has no significant impact.

7.1. Security Reason Codes

BPAs must process blocks and bundles in accordance with both BP policy and BPSec policy. The decision to receive, forward, deliver, or delete a bundle may be communicated to the report-to address of the bundle in the form of a status report, as a method of tracking the progress of the bundle through the network. The status report for a bundle may be augmented with a "reason code" explaining why the particular action was taken on the bundle.

This section describes a set of reason codes associated with the security processing of a bundle. The communication of security-related status reports might reduce the security of a network if these reports are intercepted by unintended recipients. BPSec policy **SHOULD** specify the conditions in which sending security reason codes are appropriate. Examples of appropriate conditions for the use of security reason codes could include the following.

- When the report-to address is verified as unchanged from the bundle source. This can occur by placing an appropriate BIB on the bundle primary block.
- When the block containing a status report with a security reason code is encrypted by a BCB.
- When a status report containing a security reason code is only sent for security issues relating to bundles and/or blocks associated with non-operational user data or test data.
- When a status report containing a security reason code is only sent for security issues associated with non-operational security contexts, or security contexts using non-operational configurations, such as test keys.

Security reason codes are assigned in accordance with Section 11.2 and are as described below.

Missing security operation:
: This reason code indicates that a bundle was missing one or more required security operations. This reason code is typically used by a security verifier or security acceptor.

Unknown security operation:
: This reason code indicates that one or more security operations present in a bundle cannot be understood by the security verifier or security acceptor for the operation. For example, this reason code may be used if a security block references an unknown security context identifier or security context parameter. This reason code should not be used for security

operations for which the node is not a security verifier or security acceptor; there is no requirement that all nodes in a network understand all security contexts, security context parameters, and security services for every bundle in a network.

Unexpected security operation:
: This reason code indicates that a receiving node is neither a security verifier nor a security acceptor for at least one security operation in a bundle. This reason code should not be seen as an error condition: not every node is a security verifier or security acceptor for every security operation in every bundle. In certain networks, this reason code may be useful in identifying misconfigurations of security policy.

Failed security operation:
: This reason code indicates that one or more security operations in a bundle failed to process as expected for reasons other than misconfiguration. This may occur when a security-source is unable to add a security block to a bundle. This may occur if the target of a security operation fails to verify using the defined security context at a security verifier. This may also occur if a security operation fails to be processed without error at a security acceptor.

Conflicting security operation:
: This reason code indicates that two or more security operations in a bundle are not conformant with the BPSec specification and that security processing was unable to proceed because of a BPSec protocol violation.

8. Security Considerations

Given the nature of DTN applications, it is expected that bundles may traverse a variety of environments and devices that each pose unique security risks and requirements on the implementation of security within BPSec. For this reason, it is important to introduce key threat models and describe the roles and responsibilities of the BPSec protocol in protecting the confidentiality and integrity of the data against those threats. This section provides additional discussion on security threats that BPSec will face and describes how BPSec security mechanisms operate to mitigate these threats.

The threat model described here is assumed to have a set of capabilities identical to those described by the Internet Threat Model in [RFC3552], but the BPSec threat model is scoped to illustrate threats specific to BPSec operating within DTN environments; therefore, it focuses on on-path attackers (OPAs). In doing so, it is assumed that the delay-tolerant network (or significant portions of the delay-tolerant network) are completely under the control of an attacker.

8.1. Attacker Capabilities and Objectives

BPSec was designed to protect against OPA threats that may have access to a bundle during transit from its source, Alice, to its destination, Bob. An OPA node, Olive, is a noncooperative node operating on the delay-tolerant network between Alice and Bob that has the ability to receive bundles, examine bundles, modify bundles, forward bundles, and generate bundles at will in order to compromise the confidentiality or integrity of data within the delay-tolerant network. There are three classes of OPA nodes that are differentiated based on their access to cryptographic material:

Unprivileged Node: Olive has not been provisioned within the secure environment and only has access to cryptographic material that has been publicly shared.

Legitimate Node: Olive is within the secure environment; therefore, Olive has access to cryptographic material that has been provisioned to Olive (i.e., K_M) as well as material that has been publicly shared.

Privileged Node: Olive is a privileged node within the secure environment; therefore, Olive has access to cryptographic material that has been provisioned to Olive, Alice, and/or Bob (i.e., K_M, K_A, and/or K_B) as well as material that has been publicly shared.

If Olive is operating as a privileged node, this is tantamount to compromise; BPSec does not provide mechanisms to detect or remove Olive from the delay-tolerant network or BPSec secure environment. It is up to the BPSec implementer or the underlying cryptographic mechanisms to provide appropriate capabilities if they are needed. It should also be noted that if the implementation of BPSec uses a single set of shared cryptographic material for all nodes, a legitimate node is equivalent to a privileged node because $K_M == K_A == K_B$. For this reason, sharing cryptographic material in this way is not recommended.

A special case of the legitimate node is when Olive is either Alice or Bob (i.e., $K_M == K_A$ or $K_M == K_B$). In this case, Olive is able to impersonate traffic as either Alice or Bob, respectively, which means that traffic to and from that node can be decrypted and encrypted, respectively. Additionally, messages may be signed as originating from one of the endpoints.

8.2. Attacker Behaviors and BPSec Mitigations

8.2.1. Eavesdropping Attacks

Once Olive has received a bundle, she is able to examine the contents of that bundle and attempt to recover any protected data or cryptographic keying material from the blocks contained within. The protection mechanism that BPSec provides against this action is the BCB, which encrypts the contents of its security target, providing confidentiality of the data. Of course, it should be assumed that Olive is able to attempt offline recovery of encrypted data, so the cryptographic mechanisms selected to protect the data should provide a suitable level of protection.

When evaluating the risk of eavesdropping attacks, it is important to consider the lifetime of bundles on DTN. Depending on the network, bundles may persist for days or even years. Long-lived bundles imply that the data exists in the network for a longer period of time and, thus, there may be more opportunities to capture those bundles. Additionally, the implication is that long-lived bundles store information within that remains relevant and sensitive for long enough that, once captured, there is sufficient time to crack encryption associated with the bundle. If a bundle does persist on the network for years and the cipher suite used for a BCB provides inadequate protection, Olive may be able to recover the protected data either before that bundle reaches its intended destination or before the information in the bundle is no longer considered sensitive.

NOTE: Olive is not limited by the bundle lifetime and may retain a given bundle indefinitely.

NOTE: Irrespective of whether BPSec is used, traffic analysis will be possible.

8.2.2. Modification Attacks

As a node participating in the delay-tolerant network between Alice and Bob, Olive will also be able to modify the received bundle, including non-BPSec data such as the primary block, payload blocks, or block processing control flags as defined in [RFC9171]. Olive will be able to undertake activities including modification of data within the blocks, replacement of blocks, addition of blocks, or removal of blocks. Within BPSec, both the BIB and BCB provide integrity-protection mechanisms to detect or prevent data manipulation attempts by Olive.

The BIB provides that protection to another block that is its security target. The cryptographic mechanisms used to generate the BIB should be strong against collision attacks, and Olive should not have access to the cryptographic material used by the originating node to generate the BIB (e.g., K_A). If both of these conditions are true, Olive will be unable to modify the security target or the BIB, and thus she cannot lead Bob to validate the security target as originating from Alice.

Since BPSec security operations are implemented by placing blocks in a bundle, there is no in-band mechanism for detecting or correcting certain cases where Olive removes blocks from a bundle. If Olive removes a BCB, but keeps the security target, the security target remains encrypted and there is a possibility that there may no longer be sufficient information to decrypt the block at its destination. If Olive removes both a BCB (or BIB) and its security target, there is no evidence left in the bundle of the security operation. Similarly, if Olive removes the BIB, but not the security target, there is no evidence left in the bundle of the security operation. In each of these cases, the implementation of BPSec must be combined with policy configuration at endpoints in the network that describe the expected and required security operations that must be applied on transmission and that are expected to be present on receipt. This or other similar out-of-band information is required to correct for removal of security information in the bundle.

A limitation of the BIB may exist within the implementation of BIB validation at the destination node. If Olive is a legitimate node within the delay-tolerant network, the BIB generated by Alice with K_A can be replaced with a new BIB generated with K_M and forwarded to Bob. If Bob is only validating that the BIB was generated by a legitimate user, Bob will acknowledge the message as originating from Olive instead of Alice. Validating a BIB indicates only that the BIB was generated by a holder of the relevant key; it does not provide any guarantee that the bundle or block was created by the same entity. In order to provide verifiable integrity checks, the BCB should require an encryption scheme that is Indistinguishable under adaptive Chosen Ciphertext Attack (IND-CCA2) secure. Such an encryption scheme will guard against signature substitution attempts by Olive. In this case, Alice creates a BIB with the protected data block as the security target and then creates a BCB with both the BIB and protected data block as its security targets.

8.2.3. Topology Attacks

If Olive is in an OPA position within the delay-tolerant network, she is able to influence how any bundles that come to her may pass through the network. Upon receiving and processing a bundle that must be routed elsewhere in the network, Olive has three options as to how to proceed: not forward the bundle, forward the bundle as intended, or forward the bundle to one or more specific nodes within the network.

Attacks that involve rerouting the bundles throughout the network are essentially a special case of the modification attacks described in this section, one where the attacker is modifying fields within the primary block of the bundle. Given that BPSec cannot encrypt the contents of the primary block, alternate methods must be used to prevent this situation. These methods may include requiring BIBs for primary blocks, using encapsulation, or otherwise strategically manipulating primary block data. The details of any such mitigation technique are specific to the implementation of the deploying network and are outside of the scope of this document.

Furthermore, routing rules and policies may be useful in enforcing particular traffic flows to prevent topology attacks. While these rules and policies may utilize some features provided by BPSec, their definition is beyond the scope of this specification.

8.2.4. Message Injection

Olive is also able to generate new bundles and transmit them into the delay-tolerant network at will. These bundles may be either 1) copies or slight modifications of previously observed bundles (i.e., a replay attack) or 2) entirely new bundles generated based on the Bundle Protocol, BPSec, or other bundle-related protocols. With these attacks, Olive's objectives may vary, but may be targeting either the Bundle Protocol or application-layer protocols conveyed by the Bundle Protocol. The target could also be the storage and computing capabilities of the nodes running the bundle or application-layer protocols (e.g., a denial of service to flood on the storage of the store-and-forward mechanism or a computation that would process the bundles and perhaps prevent other activities).

BPSec relies on cipher suite capabilities to prevent replay or forged message attacks. A BCB used with appropriate cryptographic mechanisms may provide replay protection under certain circumstances. Alternatively, application data itself may be augmented to include mechanisms to assert data uniqueness and then be protected with a BIB, a BCB, or both along with other block data. In such a case, the receiving node would be able to validate the uniqueness of the data.

For example, a BIB may be used to validate the integrity of a bundle's primary block, which includes a timestamp and lifetime for the bundle. If a bundle is replayed outside of its lifetime, then the replay attack will fail as the bundle will be discarded. Similarly, additional blocks, such as the Bundle Age, may be signed and validated to identify replay attacks. Finally, security context parameters within BIBs and BCBs may include anti-replay mechanisms such as session identifiers, nonces, and dynamic passwords as supported by network characteristics.

9. Security Context Considerations

9.1. Mandating Security Contexts

Because of the diversity of networking scenarios and node capabilities that may utilize BPSec, there is a risk that a single security context mandated for every possible BPSec implementation is not feasible. For example, a security context appropriate for a resource-constrained node with limited connectivity may be inappropriate for use in a well-resourced, well-connected node.

This does not mean that the use of BPSec in a particular network is meant to happen without security contexts for interoperability and default behavior. Network designers must identify the minimal set of security contexts necessary for functions in their network. For example, a default set of security contexts could be created for use over the terrestrial Internet, and they could be required by any BPSec implementation communicating over the terrestrial Internet.

To ensure interoperability among various implementations, all BPSec implementations **MUST** support at least the current, mandatory security context(s) defined in IETF Standards Track RFCs. As of this writing, that BP mandatory security context is specified in [RFC9173], but the mandatory security context(s) might change over time in accordance with usual IETF processes. Such changes are likely to occur in the future if/when flaws are discovered in the applicable cryptographic algorithms, for example.

Additionally, BPSec implementations need to support the security contexts that are required by the BP networks in which they are deployed.

If a node serves as a gateway between two or more networks, the BPSec implementation at that node needs to support the union of security contexts mandated in those networks.

BPSec has been designed to allow for a diversity of security contexts and for new contexts to be defined over time. The use of different security contexts does not change the BPSec protocol itself, and the definition of new security contexts **MUST** adhere to the requirements of such contexts as presented in this section and generally in this specification.

Implementers should monitor the state of security context specifications to check for future updates and replacement.

9.2. Identification and Configuration

Security blocks uniquely identify the security context to be used in the processing of their security services. The security context for a security block **MUST** be uniquely identifiable and **MAY** use parameters for customization.

To reduce the number of security contexts used in a network, security context designers should make security contexts customizable through the definition of security context parameters. For example, a single security context could be associated with a single cipher suite and security context parameters could be used to configure the use of this security context with different key lengths and different key management options without needing to define separate security contexts for each possible option.

A single security context may be used in the application of more than one security service. This means that a security context identifier **MAY** be used with a BIB, with a BCB, or with any other BPSec-compliant security block. The definition of a security context **MUST** identify which security services may be used with the security context, how security context parameters are interpreted as a function of the security operation being supported, and which security results are produced for each security service.

Network operators must determine the number, type, and configuration of security contexts in a system. Networks with rapidly changing configurations may define relatively few security contexts with each context customized with multiple parameters. For networks with more stability, or an increased need for confidentiality, a larger number of contexts can be defined with each context supporting few, if any, parameters.

Context Type	Parameters	Definition
Key Exchange AES	Encrypted Key, IV	AES-GCM-256 cipher suite with provided ephemeral key encrypted with a predetermined key encryption key and cleartext initialization vector.
Pre-Shared Key AES	IV	AES-GCM-256 cipher suite with predetermined key and predetermined key-rotation policy.
Out-of-Band AES	None	AES-GCM-256 cipher suite with all info predetermined.

Table 1: Security Context Examples

9.3. Authorship

Developers or implementers should consider the diverse performance and conditions of networks on which the Bundle Protocol (and, therefore, BPSec) will operate. Specifically, the delay and capacity of DTNs can vary substantially. Developers should consider these conditions to better describe the conditions in which those contexts will operate or exhibit vulnerability, and selection of these contexts for implementation should be made with consideration for this reality. There are key differences that may limit the opportunity for a security context to leverage existing cipher suites and technologies that have been developed for use in more reliable networks:

Data Lifetime: Depending on the application environment, bundles may persist on the network for extended periods of time, perhaps even years. Cryptographic algorithms should be selected to ensure protection of data against attacks for a length of time reasonable for the application.

One-Way Traffic: Depending on the application environment, it is possible that only a one-way connection may exist between two endpoints, or if a two-way connection does exist, the round-trip time may be extremely large. This may limit the utility of session key generation mechanisms, such as Diffie-Hellman, as a two-way handshake may not be feasible or reliable.

Opportunistic Access: Depending on the application environment, a given endpoint may not be guaranteed to be accessible within a certain amount of time. This may make asymmetric cryptographic architectures that rely on a key distribution center or other trust center impractical under certain conditions.

When developing security contexts for use with BPSec, the following information **SHOULD** be considered for inclusion in these specifications.

Security Context Parameters: Security contexts **MUST** define their parameter Ids, the data types of those parameters, and their CBOR encoding.

Security Results: Security contexts **MUST** define their security result Ids, the data types of those results, and their CBOR encoding.

New Canonicalizations: Security contexts may define new canonicalization algorithms as necessary.

Ciphertext Size: Security contexts **MUST** state whether their associated cipher suites generate ciphertext (to include any authentication information) that is of a different size than the input plaintext.

If a security context does not wish to alter the size of the plaintext, it should place overflow bytes and authentication tags in security result fields.

Block Header Information: Security contexts **SHOULD** include block header information that is considered to be immutable for the block. This information **MAY** include the block type code, block number, CRC type, and CRC field (if present or if missing and unlikely to be added later), and possibly certain block processing control flags. Designers should input these fields as additional data for integrity protection when these fields are expected to remain unchanged over the path the block will take from the security source to the security acceptor. Security contexts considering block header information **MUST** describe expected behavior when these fields fail their integrity verification.

Handling CRC Fields: Security contexts may include algorithms that alter the contexts of their security target block, such as the case when encrypting the block-type-specific data of a target block as part of a BCB confidentiality service. Security context specifications **SHOULD** address how preexisting CRC type and CRC value fields be handled. For example, a BCB security context could remove the plaintext CRC value from its target upon encryption and replace or recalculate the value upon decryption.

10. Defining Other Security Blocks

Other Security Blocks (OSBs) may be defined and used in addition to the security blocks identified in this specification. BIB, BCB, and any future OSBs can coexist within a bundle and can be considered in conformance with BPSec if all of the following requirements are met by any future identified security blocks.

- OSBs **MUST NOT** reuse any enumerations identified in this specification, to include the block type codes for BIB and BCB.
- An OSB definition **MUST** state whether it can be the target of a BIB or a BCB. The definition **MUST** also state whether the OSB can target a BIB or a BCB.
- An OSB definition **MUST** provide a deterministic processing order in the event that a bundle is received containing BIBs, BCBs, and OSBs. This processing order **MUST NOT** alter the BIB and BCB processing orders identified in this specification.
- An OSB definition **MUST** provide a canonicalization algorithm if the default algorithm for non-primary-block canonicalization cannot be used to generate a deterministic input for a cipher suite. This requirement can be waived if the OSB is defined so as to never be the security target of a BIB or a BCB.
- An OSB definition **MUST NOT** require any behavior of a BPSec BPA that is in conflict with the behavior identified in this specification. In particular, the security processing requirements imposed by this specification must be consistent across all BPSec BPAs in a network.
- The behavior of an OSB when dealing with fragmentation must be specified and **MUST NOT** lead to ambiguous processing states. In particular, an OSB definition should address how to receive and process an OSB in a bundle fragment that may or may not also contain its security target. An OSB definition should also address whether an OSB may be added to a bundle marked as a fragment.

Additionally, policy considerations for the management, monitoring, and configuration associated with blocks **SHOULD** be included in any OSB definition.

NOTE: The burden of showing compliance with processing rules is placed upon the specifications defining new security blocks, and the identification of such blocks shall not, alone, require maintenance of this specification.

11. IANA Considerations

This specification includes fields that require registries managed by IANA.

11.1. Bundle Block Types

This specification allocates two block types from the existing "Bundle Block Types" registry defined in [RFC6255].

Value	Description	Reference
11	Block Integrity	This document
12	Block Confidentiality	This document

Table 2: Additional Entries for the "Bundle Block Types" Registry

The "Bundle Block Types" registry notes whether a block type is meant for use in BP version 6, BP version 7 (BPv7), or both. The two block types defined in this specification are meant for use with BPv7.

11.2. Bundle Status Report Reason Codes

This specification allocates five reason codes from the existing "Bundle Status Report Reason Codes" registry defined in [RFC6255].

BP Version	Value	Description	Reference
7	12	Missing security operation	This document, Section 7.1
7	13	Unknown security operation	This document, Section 7.1
7	14	Unexpected security operation	This document, Section 7.1
7	15	Failed security operation	This document, Section 7.1
7	16	Conflicting security operation	This document, Section 7.1

Table 3: Additional Entries for the "Bundle Status Report Reason Codes" Registry

11.3. Security Context Identifiers

BPSec has a Security Context Identifier field for which IANA has created a new registry named "BPSec Security Context Identifiers". Initial values for this registry are given below.

The registration policy for this registry is Specification Required (see [RFC8126]).

The value range: signed 16-bit integer.

Value	Description	Reference
< 0	Reserved	This document
0	Reserved	This document

Table 4: "BPSec Security Context Identifier" Registry

Negative security context identifiers are reserved for local/site-specific uses. The use of 0 as a security context identifier is for nonoperational testing purposes only.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC6255] Blanchet, M., "Delay-Tolerant Networking Bundle Protocol IANA Registries", RFC 6255, DOI 10.17487/RFC6255, May 2011, <https://www.rfc-editor.org/info/rfc6255>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

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

[RFC9173] Birrane, III, E., White, A., and S. Heiner, "Default Security Contexts for Bundle Protocol Security (BPSec)", RFC 9173, DOI 10.17487/RFC9173, January 2022, <https://www.rfc-editor.org/info/rfc9173>.

12.2. Informative References

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <https://www.rfc-editor.org/info/rfc4838>.

[RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", RFC 6257, DOI 10.17487/RFC6257, May 2011, <https://www.rfc-editor.org/info/rfc6257>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

Acknowledgments

The following participants contributed technical material, use cases, and useful thoughts on the overall approach to this security specification: Scott Burleigh of the IPNGROUP, Angela Hennessy of the Laboratory for Telecommunications Sciences, Amy Alford and Cherita Corbett of the Johns Hopkins University Applied Physics Laboratory (JHU/APL), and Angela Dalton of AMD Research.

Additionally, Benjamin Kaduk of Akamai Technologies provided a detailed technical review that resulted in a stronger and more precise specification.

Authors' Addresses

Edward J. Birrane, III
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
United States of America
Phone: +1 443 778 7423
Email: Edward.Birrane@jhuapl.edu

Kenneth McKeever
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
United States of America
Phone: +1 443 778 2237
Email: Ken.McKeever@jhuapl.edu