Delay-Tolerant Networking E. Birrane
Internet-Draft JHU/APL
Intended status: Experimental J. Mayer
Expires: April 18, 2016 INSYEN AG
D. Iannicca
NASA GRC
October 16, 2015

Streamlined Bundle Security Protocol Specification
draft-birrane-dtn-sbsp-01

Abstract

This document defines a streamlined bundle security protocol, which provides data authentication, integrity, and confidentiality services for the Bundle Protocol. Capabilities are provided to protect blocks in a bundle along a single path through a network.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines security features for the Bundle Protocol [RFC5050] intended for use in delay-tolerant networks, in order to provide Delay-Tolerant Networking (DTN) security services.

The Bundle Protocol is used in DTNs that overlay multiple networks, some of which may be challenged by limitations such as intermittent and possibly unpredictable loss of connectivity, long or variable delay, asymmetric data rates, and high error rates. The purpose of the Bundle Protocol is to support interoperability across such stressed networks.

The stressed environment of the underlying networks over which the Bundle Protocol operates makes it important for the DTN to be protected from unauthorized use, and this stressed environment poses unique challenges for the mechanisms needed to secure the Bundle Protocol. Furthermore, DTNs may be deployed in environments where a portion of the network might become compromised, posing the usual security challenges related to confidentiality, integrity, and availability.

This document describes the Streamlined Bundle Security Protocol (SBSP), which provides security services for blocks within a bundle from the bundle source to the bundle destination. Specifically, the SBSP provides authentication, integrity, and confidentiality for bundles along a path through a DTN.

SBSP applies, by definition, only to those nodes that implement it, known as "security-aware" nodes. There MAY be other nodes in the DTN that do not implement SBSP. All nodes can interoperate with the exception that SBSP security operations can only happen at SBSP security-aware nodes.

1.1. 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 delay-tolerant networks, but does not discuss security at any length.

The DTN Bundle Protocol [RFC5050] defines the format and processing of the blocks used to implement the Bundle Protocol, excluding the security-specific blocks defined here.

The Bundle Security Protocol [RFC6257] introduces the concepts of security blocks for authentication, confidentiality, and integrity. The SBSP is based off of this document.

1.2. 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 [RFC2119].

We introduce the following terminology for purposes of clarity.

- Source - the bundle node from which a bundle originates.
- Destination - the bundle node to which a bundle is ultimately destined.
- Forwarder - the bundle node that forwarded the bundle on its most recent hop.
- Intermediate Receiver, Waypoint, or "Next Hop" - the neighboring bundle node to which a forwarder forwards a bundle.
- Path - the ordered sequence of nodes through which a bundle passes on its way from source to destination. The path is not necessarily known by the bundle, or any bundle-aware nodes.

Figure 1 below is adapted from [RFC5050] and shows four bundle nodes (denoted BN1, BN2, BN3, and BN4) that reside above some transport layer(s). Three distinct transport and network protocols (denoted T1/N1, T2/N2, and T3/N3) are also shown.

```
   +---------v-|   +->>>>>>>>>>v-+     +->>>>>>>>>>v-+   +-^---------+
   | BN1     v |   | ^   BN2   v |     | ^   BN3   v |   | ^  BN4    |
   +---------v-+   +-^---------v-+     +-^---------v-+   +-^---------+
   | T1      v |   + ^  T1/T2  v |     + ^  T2/T3  v |   | ^  T3     |
   +---------v-+   +-^---------v-+     +-^---------v +   +-^---------+
   | N1      v |   | ^  N1/N2  v |     | ^  N2/N3  v |   | ^  N3     |
   +---------v-+   +-^---------v +     +-^---------v-+   +-^---------+
   |         >>>>>>>>^         >>>>>>>>>>^         >>>>>>>>^         |
   +-----------+   +------------+      +-------------+   +-----------+
   |                     |                    |                      |
   |<--  An Internet --->|                    |<--- An Internet  --->|
   |                     |                    |                      |
```

Figure 1: Bundle Nodes Sit at the Application Layer of the Internet Model

BN1 originates a bundle that it forwards to BN2. BN2 forwards the bundle to BN3, and BN3 forwards the bundle to BN4. BN1 is the source of the bundle and BN4 is the destination of the bundle. BN1 is the first forwarder, and BN2 is the first intermediate receiver; BN2 then becomes the forwarder, and BN3 the intermediate receiver; BN3 then becomes the last forwarder, and BN4 the last intermediate receiver, as well as the destination.

If node BN2 originates a bundle (for example, a bundle status report or a custodial signal), which is then forwarded on to BN3, and then to BN4, then BN2 is the source of the bundle (as well as being the first forwarder of the bundle) and BN4 is the destination of the bundle (as well as being the final intermediate receiver).

We introduce the following security-specific DTN terminology.

- Security-Service - the security features supported by this specification: authentication, integrity, and confidentiality.

- Security-Source - a bundle node that adds a security block to a bundle.

- Security-Destination - a bundle node that evaluates a security block from a bundle. When a security-service is applied hop-by-hop, the security-destination is the next intermediate receiver. Otherwise, the security-destination is the same as the bundle destination.

- Security-Target - the portion of a bundle (e.g., the primary block, payload block, extension block, or entire bundle) that receives a security-service as part of a security-operation.

- Security Block - a single instance of a SBSP extension block in a bundle.

- Security-Operation - the application of a security-service to a specific security-target, notated as OP(security-service, security-target). For example, OP(authentication, bundle) or OP(confidentiality, payload). Every security-operation in a bundle MUST be unique, meaning that a security-service can only be applied to a security-target once in a bundle. A security-operation MAY be implemented by one or more security blocks.

2. Key Properties

The application of security services in a DTN is a complex endeavor that must consider physical properties of the network, policies at each node, and various application security requirements. Rather than enumerate all potential security implementations in all potential DTN topologies, this specification defines a set of key properties of a security system. The security primitives outlined in this document MUST enable the realization of these properties in a DTN deploying the Bundle Protocol.

2.1. Block-Level Granularity

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 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 may represent different types of information with different security needs.

Security services within this specification MUST provide block level granularity where applicable such that different blocks within a bundle may have different security services applied to them.

For example, within a bundle, a payload might be encrypted to protect its contents, whereas an extension block containing summary information related to the payload might be integrity signed but otherwise unencrypted to provide certain nodes access to payload-related data without providing access to the payload.

2.2. Multiple Security Sources

The Bundle Protocol allows extension blocks to be added to a bundle at any time during its existence in the DTN. When a waypoint node adds a new extension block to a bundle, that extension block may have security services applied to it by that waypoint. Similarly, a waypoint node may add a security service to an existing extension block, consistent with its security policy. For example, a node representing a boundary between a trusted part of the network and an untrusted part of the network may wish to apply payload encryption for bundles leaving the trusted portion of the network.

In each case, a node other than the bundle originator may be adding a security service to the bundle and, as such, the source for the security service will be different than the source of the bundle itself. Security services MUST track their orginating node so as to properly apply policy and key selection associated with processing the security service at the bundle destination.

Referring to Figure 1, if the bundle that originates at BN1 is given security blocks by BN1, then BN1 is the security-source for those blocks as well as being the source of the bundle. If the bundle that originates at BN1 is then given a security block by BN2, then BN2 is the security-source for that block even though BN1 remains the bundle source.

A bundle MAY have multiple security blocks and these blocks MAY have different security-sources. Each security block in a bundle will be associated with a specific security-operation. All security blocks comprising a security-operation MUST have the same security-source and security-destination.

As required in [RFC5050], forwarding nodes MUST transmit blocks in a bundle in the same order in which they were received. This requirement applies to all DTN nodes, not just ones that implement security processing. Blocks in a bundle MAY be added or deleted according to the applicable specification, but those blocks that are both received and transmitted MUST be transmitted in the same order that they were received.

2.3. Single Security Destinations

The destination of all security blocks in a bundle MUST be the bundle destination, with the exception of authentication security blocks, whose destination is the next hop along the bundle path. In a DTN, there is typically no guarantee that a bundle will visit a particular intermediate receiver during its journey, or that a particular series of intermediate receivers will be visited in a particular order.

Security-destinations different from bundle destinations would place a tight (and possibly intractable) coupling between security and routing services in an overlay network.

2.4. Mixed Security Policy

Different nodes in a DTN may have different security-related capabilities. Some nodes may not be security-aware and will not understand any security-related extension blocks. Other nodes may have security policies that require evaluation of security services at places other than the bundle destination (such as verifying integrity signatures at certain waypoint nodes). Other nodes may ignore any security processing if they are not the destination of the bundle. The security services described in this specification must allow each of these scenarios.

Extension blocks representing security services MUST have their block processing flags set such that the block (and bundle, where applicable) will be treated appropriately by non-security-aware nodes.

Extension blocks providing integrity and authentication services within a bundle MUST support options to allow waypoint nodes to evaluate these signatures if such nodes have the proper configuraton to do so.

2.5. User-Selected Ciphersuites

The security services defined in this specification rely on a a variety of ciphersuites providing integrity signatures, ciphertext, and other information necessary to populate security blocks. Users may wish to select differing ciphersuites to implement different security services. For example, some users may wish to use a SHA-1 based hash for integrity whereas other users may require a SHA-2 hash instead. The security services defined in this specification MUST provide a mechanism for identifying what ciphersuite has been used to populate a security block.

2.6. Deterministic Processing

In all cases, the processing order of security services within a bundle must avoid ambiguity when evaluating security at the bundle destination. This specification MUST provide determinism in the application and evaluation of security services, even when doing so results in a loss of flexibility.

3. Security Block Definitions

There are four types of security blocks that MAY be included in a bundle. These are the Bundle Authentication Block (BAB), the Block Integrity Block (BIB), the Block Confidentiality Block (BCB), and the Cryptographic Messaging Syntax Block (CMSB).

> The BAB is used to ensure the authenticity and integrity of the bundle along a single hop from forwarder to intermediate receiver. As such, BABs operate between topologically adjacent nodes. Security-aware nodes MAY choose to require BABs from a given neighbor in the network in order to receive and process a received bundle.
>
> The BIB is used to ensure the authenticity and integrity of its security-target from the BIB security-source, which creates the BIB, to the bundle destination, which verifies the BIB authenticator. The authentication information in the BIB MAY (when possible) be verified by any node in between the BIB security-source and the bundle destination.
>
> The BCB indicates that the security-target has been encrypted, in whole or in part, at the BCB security-source in order to protect its content while in transit to the bundle destination.
>
> The CMSB contains a Cryptographic Message Syntax (CMS) payload used to describe a security service applied to another extension block. NOTE: Applications may choose to simply place CMS text as the payload to the bundle. In such cases, security is considered to be implemented at the application layer and CMSBs are not required in that case.

Certain cipher suites may allow or require multiple instances of a block to appear in the bundle. For example, an authentication cipher suite may require two security blocks, one before the payload block and one after. Despite the presence of two security blocks, they both comprise the same security-operation - OP(authentication,bundle) in this example.

A security-operation MUST NOT be applied more than once in a bundle. For example, the two security-operations: OP(integrity, payload) and OP(integrity, payload) are considered redundant and MUST NOT appear together in a bundle. However, the two security operations OP(integrity, payload) and OP(integrity, extension_block_1) MAY both be present in the bundle. Also, the two security operations OP(integrity, extension_block_1) and OP(integrity, extension_block_2) are unique and may both appear in the same bundle.

Many of the fields in these block definitions use the Self-Delimiting Numeric Value (SDNV) type whose format and encoding is as defined in [RFC5050].

3.1. Block Identification

This specification requires that every target block of a security operation be uniquely identifiable. In cases where there can only be a single instance of a block in the bundle (as is the case with the primary block and the payload block) then the unique identifier is simply the block type. These blocks are described as "singleton blocks". It is possible that a bundle may contain multiple instances of a block type. In such a case, each instance of the block type must be uniquely identifiable and the block type itself is not sufficient for this identification. These blocks are described as "non-singleton blocks".

The definition of the extension block header from [RFC5050] does not provide additional identifying information for a block beyond the block type. The addition of an occurrence number to the block is necessary to identify the block instance in the bundle. This section describes the use of an Artificial EID (AEID) reference in a block header to add unique identification for non-singleton blocks.

Figure 7 of [RFC5050] illustrates that an EID reference in a block header is the 2-tuple of the reference scheme and the reference scheme specific part (SSP), each of which are encoded as SDNVs. The AEID MUST encode the occurrence number in the reference scheme SDNV and MUST set the reference SSP to 0. A reference SSP value of 0 is an invalid offset for an SSP in the bundle dictionary and, therefore, the use of 0 in this field identifies the reference as an AEID.

The occurrence number MAY be any positive value that is not already present as an occurrence number for the same block type in the bundle. These numbers are independent of relative block position within the bundle, and whether blocks of the same type have been added or removed from the bundle. Once an AEID has been added to a block instance, it MUST NOT be changed until all security operations that target the block instance have been removed from the bundle.

If a node wishes to apply a security operation to a target block it MUST determine whether the target block is a singleton block or a non-singleton block. If the target block is non-singleton, then the node MUST find the AEID for the target. If an AEID is not present in the target block header then the node MAY choose to either cancel the security operation or add an AEID to the block, in accordance with security policy.

If a node chooses to add an AEID to a target block header it MUST perform the following activities.

- The "Block contains an EID reference field" flag MUST be set for the target block, if it is not already set.

- The EID reference count for the block MUST be updated to reflect the addition of the AEID.

- The scheme offset of the AEID MUST be a value greater than 0. The scheme offset MUST NOT be the same as any other AEID of any other block in the bundle sharing the same block type.

- The SSP offset of the AEID MUST be the value 0. There MUST NOT be any other EID in the block header that has a value of 0 for the SSP offset.

If there is no AEID present in a block, and if a node is unable to add an AEID by following the above process, then the block MUST NOT have an SBSP security operation applied to it.

It is RECOMMENDED that every block in a bundle other than the primary and payload blocks be treated as a non-singleton block. However, the identification of singleton blocks SHOULD be in accordance with the security policy of a node.

3.2. Abstract Security Block

Each security block uses the Canonical Bundle Block Format as defined in [RFC5050]. That is, each security block is comprised of the following elements:

- Block Type Code

- Block Processing Control Flags

- Block EID Reference List (OPTIONAL)

- Block Data Length

- Block Type Specific Data Fields

Since the four security block types have most fields in common, we can shorten the description of the block type specific data fields if we first define an abstract security block (ASB) and then specify each of the real blocks in terms of the fields that are present/ absent in an ASB. Note that no bundle ever contains an actual ASB, which is simply a specification artifact.

The structure of an Abstract Security Block is given in Figure 2. Although the diagram hints at a fixed-format layout, this is purely for the purpose of exposition. Except for the "type" field, all fields are variable in length.

```
+------------------------------+----------------------------------+
|    Block Type Code (BYTE)    | Processing Control Flags (SDNV)  |
+------------------------------+----------------------------------+
|        EID Reference Count and List (Compound List)             |
+------------------------------+----------------------------------+
|    Block Length (SDNV)       |    Security Target (Compound)    |
+------------------------------+----------------------------------+
|    Cipher suite ID (SDNV)    |     Cipher suite Flags (SDNV)    |
+------------------------------+----------------------------------+
|    Params Length (SDNV)      |      Params Data (Compound)      |
+------------------------------+----------------------------------+
|    Result Length (SDNV)      |      Result Data (Compound)      |
+------------------------------+----------------------------------+
```

Figure 2: Abstract Security Block Structure

An ASB consists of the following fields, some of which are optional.

o Block-Type Code (Byte) - as described in [RFC5050]. The block-type codes for security blocks are:

   * BundleAuthenticationBlock - BAB: 0x02

   * BlockIntegrityBlock - BIB: 0x03

   * BlockConfidentialityBlock - BCB: 0x04

o Block Processing Control Flags (SDNV) - as described in [RFC5050]. There are no general constraints on the use of the block processing control flags, and some specific requirements are discussed later.

o (OPTIONAL) EID Reference Count and List - as described in [RFC5050]. Presence of the EID reference field is indicated by the setting of the "Block contains an EID reference field" (EID_REF) bit of the block processing control flags. If no EID fields are present, then the composite field itself MUST be omitted entirely and the EID_REF bit MUST be unset. A count field of zero is not permitted. The possible EIDs are:

   (OPTIONAL) Security-source - specifies the security-source for the block. If this is omitted, then the source of the bundle is assumed to be the security-source unless otherwise indicated by policy or associated cipher suite definition. When present, the security-source MUST be the first EID in the list.

(OPTIONAL) AEID - specifies an identifier that can be used to uniquely identify an instance of a non-singleton block. This field MUST be present for non-singleton blocks. This field MUST NOT be present for singleton blocks, such as the primary block and the payload block. The construction of the AEID is discussed in Section 3.1.

- Block Length (SDNV) - as described in [RFC5050].

- Block type specific data fields as follows:

  - Security-Target (Compound) - Uniquely identifies the target of the associated security-operation.

    As discussed in Section 3.1 a singleton block is identified by its block type and a non-singleton block is identified by the combination of its block type and an occurrence number. The security-target is a compound field that contains the block type (as a byte) and occurrence number (as an SDNV).

    The occurrence number of a singleton block MUST be set to 0. The occurrence number of a non-singleton block MUST be set to the scheme offset of the AEID associated with the block being targeted by the security operation.

  - (OPTIONAL) Cipher suite ID (SDNV)

  - (OPTIONAL) Cipher suite flags (SDNV)

  - (OPTIONAL) Cipher Suite Parameters - compound field of the next two items.

    - Cipher suite parameters length (SDNV) - specifies the length of the next field, which is the cipher suite-parameters data field.

    - Cipher suite parameters data - parameters to be used with the cipher suite in use, e.g., a key identifier or initialization vector (IV). See Section 3.9 for a list of potential parameters and their encoding rules. The particular set of parameters that is included in this field is defined as part of a cipher suite specification.

  - (OPTIONAL) Security Result - compound field of the next two items.

- Security result length (SDNV) - contains the length of the next field, which is the security-result data field.

- Security result data - contains the results of the appropriate cipher suite specific calculation (e.g., a signature, Message Authentication Code (MAC), or cipher-text block key).

The structure of the cipher suite flags field is shown in Figure 3. In each case, the presence of an optional field is indicated by setting the value of the corresponding flag to one. A value of zero indicates the corresponding optional field is missing. Presently, there are three flags defined for the field; for convenience, these are shown as they would be extracted from a single-byte SDNV. Future additions may cause the field to grow to the left so, as with the flags fields defined in [RFC5050], the description below numbers the bit positions from the right rather than the standard RFC definition, which numbers bits from the left.

bits 6-3 are reserved for future use.

src - bit 2 indicates whether the EID-reference field of the ASB contains the optional reference to the security-source.

parm - bit 1 indicates whether or not the cipher suite parameters length and cipher suite parameters data fields are present.

res - bit 0 indicates whether or not the ASB contains the security-result length and security-result data fields.

```
  Bit   Bit   Bit   Bit   Bit   Bit   Bit
   6     5     4     3     2     1     0
+-----+-----+-----+-----+-----+-----+-----+
|    reserved           | src |parm | res |
+-----+-----+-----+-----+-----+-----+-----+
```

Figure 3: Cipher Suite Flags

3.3. Block Ordering

A security-operation may be implemented in a bundle using either one or two security blocks. For example, the operation OP(authentication, bundle) MAY be accomplished by a single BAB block in the bundle, or it MAY be accomplished by two BAB blocks in the bundle. To avoid confusion, we use the following terminology to identify the block or blocks comprising a security-operation.

The terms "First" and "Last" are used ONLY when describing multiple security blocks comprising a single security-operation. A "First" block refers to the security block that is closest to the primary block in the canonical form of the bundle. A "Last" block refers to the security block that is furthest from the primary block in the canonical form of the bundle.

If a single security block implements the security-operation, then it is referred to as a "Lone" block. For example, when a bundle authentication cipher suite requires a single BAB block we refer to it as a Lone BAB. When a bundle authentication cipher suite requires two BAB blocks we refer to them as the First BAB and the Last BAB.

This specification and individual cipher suites impose restrictions on what optional fields must and must not appear in First blocks, Last blocks, and Lone blocks.

3.4. Bundle Authentication Block

This section describes typical field values for the BAB, which is solely used to implement OP(authentication, bundle).

The block-type code field value MUST be 0x02.

The block processing control flags value can be set to whatever values are required by local policy. Cipher suite 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.

The security-target MUST be the entire bundle, which MUST be represented by a <block type><occurrence number> of <0x00><0x00>.

The cipher suite ID MUST be documented as a hop-by-hop authentication cipher suite. When a Lone BAB is used, the cipher suite MUST be documented as requiring one instance of the BAB. When a First BAB and Last BAB are used, the cipher suite MUST be documented as requiring two instances of the BAB.

The cipher suite parameters field MAY be present, if so specified in the cipher suite specification.

An EID-reference to the security-source MAY be present in either a First BAB or a Lone BAB. An EID-reference to the security-source MUST NOT be present in a Last BAB.

The security-result captures the result of applying the cipher suite calculation (e.g., the MAC or signature) to the relevant parts of the bundle, as specified in the cipher suite definition. This field MUST be present in either a Lone BAB or a Last BAB. This field MUST NOT be present in a First BAB.

Notes:

- When multiple BAB blocks are used, the mandatory fields of the Last BAB must match those of the First BAB.
- The First BAB or Lone BAB, when present, SHOULD immediately follow the primary block.
- A Last BAB, when present, SHOULD be the last block in the bundle.
- Since OP(authentication, bundle) is allowed only once in a bundle, it is RECOMMENDED that users wishing to support multiple authentication signatures define a multi-target cipher suite, capturing multiple security results in cipher suite parameters.

3.5. Block Integrity Block

A BIB is an ASB with the following additional restrictions:

The block-type code value MUST be 0x03.

The block processing control flags value can be set to whatever values are required by local policy. Cipher suite 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.

The security-target MUST uniquely identify a block within the bundle. The reserved block type 0x01 specifies the singleton payload block. The reserved type 0x00 specifies the singleton primary block. The security-target for a BIB MUST NOT reference a security block defined in this specification (BAB, BIB, or BCB).

The cipher suite ID MUST be documented as an end-to-end authentication-cipher suite or as an end-to-end error-detection-cipher suite.

The cipher suite parameters field MAY be present in either a Lone BIB or a First BIB. This field MUST NOT be present in a Last BIB.

An EID-reference to the security-source MAY be present in either a Lone BIB or a First BIB. This field MUST NOT be present in a Last BIB.

The security-result captures the result of applying the cipher suite calculation (e.g., the MAC or signature) to the relevant parts of the security-target, as specified in the cipher suite definition. This field MUST be present in either a Lone BIB or a Last BIB. This field MUST NOT be present in a First BIB.

The cipher suite MAY process less than the entire security-target. If the cipher suite processes less than the complete, original security-target, the cipher suite parameters MUST specify which bytes of the security-target are protected.

Notes:

- Since OP(integrity, target) is allowed only once in a bundle per target, it is RECOMMENDED that users wishing to support multiple integrity signatures for the same target define a multi-signature cipher suite, capturing multiple security results in cipher suite parameters.

- For some cipher suites, (e.g., those using asymmetric keying to produce signatures or those using symmetric keying with a group key), the security information MAY be checked at any hop on the way to the destination that has access to the required keying information, in accordance with Section 3.8.

- The use of a generally available key is RECOMMENDED if custodial transfer is employed and all nodes SHOULD verify the bundle before accepting custody.

3.6. Block Confidentiality Block

A BCB is an ASB with the following additional restrictions:

The block-type code value MUST be 0x04.

The block processing control flags value can be set to whatever values are required by local policy, except that a Lone BCB or First BCB MUST have the "replicate in every fragment" flag set. This indicates to a receiving node that the payload portion in each fragment represents cipher-tex

t. This flag SHOULD NOT be set otherwise. Cipher suite 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.

The security-target MUST uniquely identify a block within the bundle. The security-target for a BCB MAY reference the payload block, a non-security extension block, or a BIB block. The reserved type 0x01 specifies the singleton payload block.

The cipher suite ID MUST be documented as a confidentiality cipher suite.

Key-information, if available, MUST appear only in a Lone BCB or a First BCB.

Any additional bytes generated as a result of encryption and/or authentication processing of the security-target SHOULD be placed in an "integrity check value" field (see Section 3.9) in the security-result of the Lone BCB or Last BCB.

The cipher suite parameters field MAY be present in either a Lone BCB or a First BCB. This field MUST NOT be present in a Last BCB.

An EID-reference to the security-source MAY be present in either a Lone BCB or a First BCB. This field MUST NOT be present in a Last BCB. The security-source can also be specified as part of key-information described in Section 3.9.

The security-result MAY be present in either a Lone BCB or a Last BCB. This field MUST NOT be present in a First BCB. This compound field normally contains fields such as an encrypted bundle encryption key and/or authentication tag.

The BCB is the only security block that modifies the contents of its security-target. When a BCB is applied, the security-target body data are encrypted "in-place". Following encryption, the security-target body data contains cipher-text, not plain-text. Other security-target block fields (such as type, processing control flags, and length) remain unmodified.

Fragmentation, reassembly, and custody transfer are adversely affected by a change in size of the payload due to ambiguity about what byte range of the block is actually in any particular fragment. Therefore, when the security-target of a BCB is the bundle payload, the BCB MUST NOT alter the size of the payload block body data. Cipher suites SHOULD place any block expansion, such as authentication tags (integrity check values) and any padding generated by a block-mode cipher, into an integrity check value item in the security-result field (see Section 3.9) of the BCB. This "in-place" encryption allows fragmentation, reassembly, and custody transfer to operate without knowledge of whether or not encryption has occurred.

Notes:

- The cipher suite MAY process less than the entire original security-target body data. If the cipher suite processes less than the complete, original security-target body data, the BCB for that security-target MUST specify, as part of the cipher suite parameters, which bytes of the body data are protected.

- The BCB’s "discard" flag may be set independently from its security-target’s "discard" flag. Whether or not the BCB’s "discard" flag is set is an implementation/policy decision for the encrypting node. (The "discard" flag is more properly called the "Discard if block cannot be processed" flag.)

- A BCB MAY include information as part of additional authenticated data to address parts of the target block, such as EID references, that are not converted to cipher-text.

3.7. Cryptographic Message Syntax Block

A CMSB is an ASB with the following additional restrictions:

The block-type code value MUST be 0x05.

The content of the block must contain valid CMS data, as defined in RFC 5652, and encoded in X.690 BER or DER encoding.

The block processing control flags value can be set to whatever values are required by local policy. This flag SHOULD NOT be set otherwise. Cipher suite 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.

The security-target MUST uniquely identify a block within the bundle. The reserved block type 0x01 specifies the singleton payload block.

The security operation(s) will be performed on the security-target block’s data and the resulting CMS content will be stored within the CMSB block’s security-result field. The security-target block’s data will then be removed.

A CMSB block MAY include multiple CMS security operations within a single block to allow for multiple nested operations to be performed on a bundle block. Multiple CMSB blocks MAY be included in a bundle as long as the security-target for each is unique.

Key-information, if available, MUST appear within the CMS content contained in the security-result field.

A CMSB block is created with its corresponding security-target field pointing to a unique bundle block. The CMS security operations are performed upon the security-target’s data field and the resulting encoded CMS content is stored within the CMS security-result field of the CMSB’s payload. The security-target block’s data MAY be left intact, replaced with alternate data, or completely erased based on the specification of the utilized CMS ciphersuite definition and applicable policy.

Multiple CMS operations may be nested within a single CMSB block to allow more than one security operation to be performed upon a security-target.

CMS Operations can be considered to have SBSP parallels: CMSB Enveloped-Data content type SHALL be considered as equivalent to a SBSP BCB block, and a CMSB Signed-Data type SHALL be considered as equivalent to a SBSP BIB block.

3.8. Block Interactions

The four 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 creating processing dependencies.

If confidentiality is being applied to a target that already has integrity applied to it, then an undesirable condition occurs where a security-aware intermediate node would be unable to check the integrity result of a block because the block contents have been encrypted after the integrity signature was generated. To address this concern, the following processing rules MUST be followed.

- If confidentiality is to be applied to a target, it MUST also be applied to every integrity operation already defined for that target. This means that if a BCB is added to encrypt a block, another BCB MUST also be added to encrypt a BIB also targeting that block.

- An integrity operation MUST NOT be applied to a security-target if a BCB in the bundle shares the same security-target. This prevents ambiguity in the order of evaluation when receiving a BIB and a BCB for a given security-target.

- An integrity value MUST NOT be evaluated if the BIB providing the integrity value is the security target of an existing BCB block in the bundle. In such a case, the BIB data contains cipher-text as it has been encrypted.

- An integrity value MUST NOT be evaluated if the security-target of the BIB is also the security-target of a BCB in the bundle. In such a case, the security-target data contains cipher-text as it has been encrypted.

- As mentioned in Section 3.6, a BIB MUST NOT have a BCB as its security target. BCBs may embed integrity results as part of cipher suite parameters.

- As mentioned in Section 4.4, CMS operations are considered to have operational parallels. When a CMSB is used, these parallels MUST be considered for block interactions (e.g., a Signed-Data structure MUST NOT be evaluated if the security-target of the operation is also the security-target of a BCB)

- If a single bundle is going to contain a CMSB as well as other security blocks, the CMS operations MUST be performed and the CMSB MUST be created before any other security operation is applied.

- On reception of a bundle containing a CMSB and other security blocks, the CMSB must be decoded last.

Additionally, since the CMSB block may contain either integrity or confidentiality information in its encapsulated CMS, there is no way to evaluate conflicts when a BIB/BCB and a CMSB have the same security target. To address this concern, the following processing rules MUST be followed.

- If an extension block is the target of a BIB or a BCB, then the extension block MUST NOT also be the target of a CMSB, and vice-versa.

- If a bundle is the target of a BAB block, then the bundle MUST NOT also be the target of a CMSB, and vice-versa.

- Generally, a CMSB MUST be processed before any BIB or BCB blocks are processed.

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, and BABs MUST be added after all other security blocks. This ordering MUST be preserved in cases where the current BPA is adding all of the security blocks for the bundle or whether the BPA is a waypoint adding new security blocks to a bundle that already contains security blocks.

3.9. Parameters and Result Fields

Various cipher suites include several items in the cipher suite parameters and/or security-result fields. Which items MAY appear is defined by the particular cipher suite description. A cipher suite MAY support several instances of the same type within a single block.

Each item is represented as a type-length-value. Type is a single byte indicating the item. Length is the count of data bytes to follow, and is an SDNV-encoded integer. Value is the data content of the item.

Item types, name, and descriptions are defined as follows.

Cipher suite parameters and result fields.

Type	Name	Description
0	Reserved	
1	Initialization Vector (IV)	A random value, typically eight to sixteen bytes.
2	Reserved	
3	Key Information	Material encoded or protected by the key management system and used to transport an ephemeral key protected by a long-term key.
4	Content Range	Pair of SDNV values (offset,length) specifying the range of payload bytes to which an operation applies. The offset MUST be the offset within the original bundle, even if the current bundle is a fragment.
5	Integrity Signatures	Result of BAB or BIB digest or other signing operation.
6	Unassigned	
7	Salt	An IV-like value used by certain confidentiality suites.
8	BCB Integrity Check Value (ICV) / Authentication Tag	Output from certain confidentiality cipher suite operations to be used at the destination to verify that the protected data has not been modified. This value MAY contain padding if required by the cipher suite.
9-255	Reserved	

Table 1

3.10. BSP Block Example

An example of SBSP blocks applied to a bundle is illustrated in Figure 4. In this figure the first column represents blocks within a bundle and the second column represents a unique identifier for each block, suitable for use as the security-target of a SBSP security-block. Since the mechanism and format of a security-target is not specified in this document, the terminology B1...Bn is used to identify blocks in the bundle for the purposes of illustration.

```
            Block in Bundle            ID
   +=================================+====+
   |         Primary Block           | B1 |
   +---------------------------------+----+
   |          First BAB              | B2 |
   |   OP(authentication, Bundle)    |    |
   +---------------------------------+----+
   |          Lone BIB               | B3 |
   |  OP(integrity, target=B1)       |    |
   +---------------------------------+----+
   |          Lone BCB               | B4 |
   |  OP(confidentiality, target=B5) |    |
   +---------------------------------+----+
   |      Extension Block            | B5 |
   +---------------------------------+----+
   |          Lone BIB               | B6 |
   |  OP(integrity, target=B7)       |    |
   +---------------------------------+----+
   |      Extension Block            | B7 |
   +---------------------------------+----+
   |          Lone BCB               | B8 |
   |  OP(confidentiality, target=B9) |    |
   +---------------------------------+----+
   |   Lone BIB  (encrypted by B8)   | B9 |
   |  OP(integrity, target=B11)      |    |
   +---------------------------------+----+
   |          Lone BCB               |B10 |
   | OP(confidentiality, target=B11) |    |
   +---------------------------------+----+
   |         Payload Block           |B11 |
   +---------------------------------+----+
   |          Last BAB               |B12 |
   |   OP(authentication, Bundle)    |    |
   +---------------------------------+----+
```

Figure 4: Sample Use of BSP Blocks

In this example a bundle has four non-security-related blocks: the primary block (B1), two extension blocks (B5,B7), and a payload block (B11). The following security applications are applied to this bundle.

- Authentication over the bundle. This is accomplished by two BAB blocks: B2 and B12.

- An integrity signature applied to the canonicalized primary block. This is accomplished by a single BIB, B3.

- Confidentiality for the first extension block. This is accomplished by a single BCB block, B4.

- Integrity for the second extension block. This is accomplished by a single BIB block, B6.

- An integrity signature on the payload. This is accomplished by a single BIB block, B9.

- Confidentiality for the payload block and it’s integrity signature. This is accomplished by two Lone BCB blocks: B8 encrypting B9, and B10 encrypting B11.

```
           Block in Bundle                     ID
   +=========================================+====+
   |         Primary Block                   | B1 |
   +-----------------------------------------+----+
   |          First BAB                      | B2 |
   |   OP(authentication, Bundle)            |    |
   +-----------------------------------------+----+
   |          Lone CMSB                      | B3 |
   |     security-target=0x01                |    |
   |     security-result=                    |    |
   |                                         |    |
   |  Signed-Data {                          |    |
   |   Digest Algorithm(s),                  |    |
   |   Enveloped-Data {                      |    |
   |     Encrypted Data,                     |    |
   |     Encrypted Encryption Key(s)         |    |
   |   },                                    |    |
   |   Signature(s) and Certificate Chain(s) |    |
   |  }                                      |    |
   |                                         |    |
   +-----------------------------------------+----+
   |         Payload Block                   | B4 |
   |        (Empty Data Field)               |    |
   +-----------------------------------------+----+
   |          Last BAB                       | B5 |
   |   OP(authentication, Bundle)            |    |
   +-----------------------------------------+----+
```

Figure 5: Sample Bundle With CMS Block

In this example a bundle has two non-security-related blocks: the primary block (B1) and a payload block (B4). This method would allow for the bundle to carry multiple CMS payloads by utilizing a multiple CMSB ASBs. The following security applications are applied to this bundle.

- Authentication over the bundle. This is accomplished by two BAB blocks: B2 and B5.

- Encrypted and signed CMS content contained within the CMSB block. The first CMS operation, encryption, is performed on the data contained within the block the security-target points to, in this case, the payload block. The resulting encrypted data is then signed and the final CMS content is stored within the CMSB block’s security-result field. The payload block’s data is subsequently removed now that the original data has been encoded within the CMSB block.

4. Security Processing

This section describes the security aspects of bundle processing.

4.1. Canonical Forms

In order to verify a signature of a bundle, the exact same bits, in the exact same order, MUST be input to the calculation upon verification as were input upon initial computation of the original signature value. Consequently, a node MUST NOT change the encoding of any URI [RFC3986] in the dictionary field, e.g., changing the DNS part of some HTTP URL from lower case to upper case. Because bundles MAY be modified while in transit (either correctly or due to implementation errors), canonical forms of security-targets MUST be defined.

Many fields in various blocks are stored as variable-length SDNVs. These are canonicalized into an "unpacked form" as eight-byte fixed-width fields in network byte order. The size of eight bytes is chosen because implementations MAY handle larger SDNV values as invalid, as noted in [RFC5050].

4.1.1. Bundle Canonicalization

Bundle canonicalization permits no changes at all to the bundle between the security-source and the destination, with the exception of one of the Block Processing Control Flags, as described below. It is intended for use in BAB cipher suites. This algorithm conceptually catenates all blocks in the order presented, but omits all security-result data fields in security blocks having the bundle as their security-target. For example, when a BAB cipher suite specifies this algorithm, we omit the BAB security-result from the catenation. The inclusion of security-result length fields is as determined by the specified cipher suite. A security-result length field MAY be present even when the corresponding security-result data fields are omitted.

Notes:

- In the Block Processing Control Flags field the unpacked SDNV is ANDed with mask 0xFFFF FFFF FFFF FFDF to zero the flag at bit 5 ("Block was forwarded without being processed"). If this flag is not zeroed out, then a bundle passing through a non-security aware node will set this flag which will change the message digest and the BAB block will fail to verify.

- In the above, we specify that security-result data is omitted. This means that no bytes of the security-result data are input.

If the security-result length is included in the catenation, we assume that the security-result length will be known to the module that implements the cipher suite before the security-result is calculated, and require that this value be in the security-result length field even though the security-result data itself will be omitted.

- The ’res’ bit of the cipher suite ID, which indicates whether or not the security-result length and security-result data field are present, is part of the canonical form.

- The value of the block data length field, which indicates the length of the block, is also part of the canonical form. Its value indicates the length of the entire block when the block includes the security-result data field.

4.1.2. Block Canonicalization

This algorithm protects those parts of a block that SHOULD NOT be changed in transit.

There are three types of blocks that may undergo block canonicalization: the primary block, the payload block, or an extension block.

4.1.2.1. Primary Block Canonicalization

The canonical form of the primary block is shown in Figure 6. Essentially, it de-references the dictionary block, adjusts lengths where necessary, and ignores flags that may change in transit.

```
   +----------------+----------------+----------------+----------------+
   |    Version     |      Processing flags (incl. COS and  SRR)       |
   +----------------+----------------+---------------------------------+
   |                Canonical primary block length                     |
   +----------------+----------------+---------------------------------+
   |                Destination endpoint ID length                     |
   +----------------+----------------+---------------------------------+
   |                      Destination endpoint ID                      |
   +----------------+----------------+---------------------------------+
   |                    Source endpoint ID length                      |
   +----------------+----------------+----------------+----------------+
   |                        Source endpoint ID                         |
   +----------------+----------------+---------------------------------+
   |                  Report-to endpoint ID length                     |
   +----------------+----------------+----------------+----------------+
   |                      Report-to endpoint ID                        |
   +----------------+----------------+----------------+----------------+
   +                    Creation Timestamp (2 x SDNV)                  +
   +---------------------------------+----------------------------------+
   |                             Lifetime                              |
   +----------------+----------------+----------------+----------------+
```

Figure 6: The Canonical Form of the Primary Bundle Block

The fields shown in Figure 6 are as follows:

- The version value is the single-byte value in the primary block.

- The processing flags value in the primary block is an SDNV, and includes the class-of-service (COS) and status report request (SRR) fields. For purposes of canonicalization, the unpacked SDNV is ANDed with mask 0x0000 0000 0007 C1BE to set to zero all reserved bits and the "bundle is a fragment" bit.

- The canonical primary block length value is a four-byte value containing the length (in bytes) of this structure, in network byte order.

- The destination endpoint ID length and value are the length (as a four-byte value in network byte order) and value of the destination endpoint ID from the primary bundle block. The URI is simply copied from the relevant part(s) of the dictionary block and is not itself canonicalized. Although the dictionary entries contain "null-terminators", the null-terminators are not included in the length or the canonicalization.

- The source endpoint ID length and value are handled similarly to the destination.

- The report-to endpoint ID length and value are handled similarly to the destination.

- The unpacked SDNVs for the creation timestamp and lifetime are copied from the primary block.

- Fragment offset and total application data unit length are ignored, as is the case for the "bundle is a fragment" bit mentioned above. If the payload data to be canonicalized is less than the complete, original bundle payload, the offset and length are specified in the cipher suite parameters.

4.1.2.2. Payload Block Canonicalization

When canonicalizing the payload block, the block processing control flags value used for canonicalization is the unpacked SDNV value with reserved and mutable bits masked to zero. The unpacked value is ANDed with mask 0x0000 0000 0000 0077 to zero reserved bits and the "last block" bit. The "last block" bit is ignored because BABs and other security blocks MAY be added for some parts of the journey but not others, so the setting of this bit might change from hop to hop.

Payload blocks are canonicalized as-is, with the exception that, in some instances, only a portion of the payload data is to be protected. In such a case, only those bytes are included in the canonical form, and additional cipher suite parameters are required to specify which part of the payload is protected, as discussed further below.

4.1.2.3. Extension Block Canonicalization

When canonicalizing an extension block, the block processing control flags value used for canonicalization is the unpacked SDNV value with reserved and mutable bits masked to zero. The unpacked value is ANDed with mask 0x0000 0000 0000 0057 to zero reserved bits, the "last block" flag and the "Block was forwarded without being processed" bit. The "last block" flag is ignored because BABs and other security blocks MAY be added for some parts of the journey but not others, so the setting of this bit might change from hop to hop.

The "Block was forwarded without being processed" flag is ignored because the bundle may pass through nodes that do not understand that extension block and this flag would be set.

Endpoint ID references in blocks are canonicalized using the de-referenced text form in place of the reference pair. The reference count is not included, nor is the length of the endpoint ID text.

The EID reference is, therefore, canonicalized as <scheme>:<SSP>, which includes the ":" character.

Since neither the length of the canonicalized EID text nor a null-terminator is used in EID canonicalization, a separator token MUST be used to determine when one EID ends and another begins. When multiple EIDs are canonicalized together, the character "," SHALL be placed between adjacent instances of EID text.

The block-length is canonicalized as its unpacked SDNV value. If the data to be canonicalized is less than the complete, original block data, this field contains the size of the data being canonicalized (the "effective block") rather than the actual size of the block.

4.1.3. Considerations

- The canonical forms for the bundle and various extension blocks is not transmitted. It is simply an artifact used as input to digesting.

- We omit the reserved flags because we cannot determine if they will change in transit. The masks specified above will have to be revised if additional flags are defined and they need to be protected.

- Our URI encoding does not preserve the null-termination convention from the dictionary field, nor do we canonicalize the scheme and scheme-specific part (SSP) separately. Instead, the byte array < scheme name > : < scheme-specific part (SSP)> is used in the canonicalization.

- The URI encoding will cause errors if any node rewrites the dictionary content (e.g., changing the DNS part of an HTTP URL from lower case to upper case). This could happen transparently when a bundle is synched to disk using one set of software and then read from disk and forwarded by a second set of software. Because there are no general rules for canonicalizing URIs (or IRIs), this problem may be an unavoidable source of integrity failures.

- All SDNV fields here are canonicalized as eight-byte unpacked values in network byte order. Length fields are canonicalized as four-byte values in network byte order. Encoding does not need optimization since the values are never sent over the network.

- These canonicalization algorithms assume that endpoint IDs themselves are immutable and they are unsuitable for use in environments where that assumption might be violated.

- Cipher suites MAY define their own canonicalization algorithms and require the use of those algorithms over the ones provided in this specification.

4.2. Endpoint ID Confidentiality

Every bundle has a primary block that contains the source and destination endpoint IDs, and possibly other EIDs (in the dictionary field) that cannot be encrypted. If endpoint ID confidentiality is required, then bundle-in-bundle encapsulation can solve this problem in some instances.

Similarly, confidentiality requirements MAY also apply to other parts of the primary block (e.g., the current-custodian), and that is supported in the same manner.

4.3. Bundles Received from Other Nodes

Security blocks MUST be processed in a specific order when received by a security-aware node. The processing order is as follows.

- All BAB blocks in the bundle MUST be evaluated prior to evaluating any other block in the bundle.

- All BCB blocks in the bundle MUST be evaluated prior to evaluating any BIBs in the bundle. When BIBs and BCBs share a security-target, BCBs MUST be evaluated first and BIBs second.

4.3.1. Receiving BAB Blocks

Nodes implementing this specification SHALL consult their security policy to determine whether or not a received bundle is required by policy to include a BAB.

If the bundle is not required to have a BAB then BAB processing on the received bundle is complete, and the bundle is ready to be further processed for BIB/BCB handling or delivery or forwarding. Security policy may provide a means to override this default behavior and require processing of a BAB if it exists.

If the bundle is required to have a BAB but does not, then the bundle MUST be discarded and processed no further. If the bundle is required to have a BAB but the key information for the security-source cannot be determined or the security-result value check fails, then the bundle has failed to authenticate, and the bundle MUST be discarded and processed no further.

If the bundle is required to have a BAB, and a BAB exists, and the BAB information is verified, then the BAB processing on the received bundle is complete, and the bundle is ready to be further processed for BIB/BCB handling or delivery or forwarding.

A BAB received in a bundle MUST be stripped before the bundle is forwarded. A new BAB MAY be added as required by policy. This MAY require correcting the "last block" field of the to-be-forwarded bundle.

4.3.2. Receiving BCB Blocks

If the bundle has a BCB and the receiving node is the destination for the bundle, the node MUST decrypt the relevant parts of the security-target in accordance with the cipher suite specification.

If the relevant parts of an encrypted payload cannot be decrypted (i.e., the decryption key cannot be deduced or decryption fails), then the bundle MUST be discarded and processed no further; in this case, a bundle deletion status report (see [RFC5050]) indicating the decryption failure MAY be generated. If any other encrypted security-target cannot be decrypted then the associated security-target and all security blocks associated with that target MUST be discarded and processed no further.

When a BCB is decrypted, the recovered plain-text MUST replace the cipher-text in the security-target body data

4.3.3. Receiving BIB Blocks

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 and, therefore, the BIB cannot be verified until the security-target has been decrypted by processing the BCB.

If the security policy of a security-aware node specifies that a bundle SHOULD apply 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. This MAY involve removing the security-target from the bundle. If the removed security-target is the payload or primary block, the bundle MAY be discarded. This action may occur at any node that has the ability to verify an integrity signature, not just the bundle destination.

If the bundle has a BIB and the receiving node is the destination for the bundle, the node MUST verify the security-target in accordance with the cipher suite specification. If a BIB check fails, the security-target has failed to authenticate and the security-target SHALL be processed according to the security policy. A bundle status report indicating the failure MAY be generated. Otherwise, if the BIB verifies, the security-target is ready to be processed for delivery.

If the bundle has a BIB and the receiving node is not the bundle destination, the receiving node MAY attempt to verify the value in the security-result field. If the check fails, the node SHALL process the security-target in accordance to local security policy. It is RECOMMENDED that if a payload integrity check fails at a waypoint that it is processed in the same way as if the check fails at the destination.

4.4. Receiving CMSB Blocks

A CMSB MUST NOT be processed if its security target is also the security target of any BAB, BIB, or BCB in the bundle.

The security services provided by a CMSB will be considered successful if all services in the CMSB are validated. If any one service encapsulated in the CMSB fails to validate, then the CMSB MUST be considered as having failed to validate and MUST be dispositioned in accordance with security policy.

4.5. Bundle Fragmentation and Reassembly

If it is necessary for a node to fragment a bundle and security services have been applied to that bundle, the fragmentation rules described in [RFC5050] MUST be followed. As defined there and repeated here for completeness, only the payload may be fragmented; security blocks, like all extension blocks, can never be fragmented. In addition, the following security-specific processing is REQUIRED:

- Due to the complexity of bundle fragmentation, including the possibility of fragmenting bundle fragments, integrity and confidentiality operations are not to be applied to a bundle fragment. Specifically, a BCB or BIB MUST NOT be added to a bundle fragment, even if the security-target of the security block is not the payload. When integrity and confidentiality must be applied to a fragment, we RECOMMEND that encapsulation be used instead.

- The authentication security policy requirements for a bundle MUST be applied individually to all the bundles resulting from a fragmentation event.

- A BAB cipher suite MAY specify that it only applies to non-fragmented bundles and not to bundle fragments.

- The decision to fragment a bundle MUST be made prior to adding authentication to the bundle. The bundle MUST first be fragmented and authentication applied to each individual fragment.

- If a bundle with a BAB is fragmented by a non-security-aware node, then the entire bundle must be re-assembled before being processed to allow for the proper verification of the BAB.

4.6. Reactive Fragmentation

When a partial bundle has been received, the receiving node SHALL consult its security policy to determine if it MAY fragment the bundle, converting the received portion into a bundle fragment for further forwarding. Whether or not reactive fragmentation is permitted SHALL depend on the security policy and the cipher suite used to calculate the BAB authentication information, if required.

Specifically, if the security policy does not require authentication, then reactive fragmentation MAY be permitted. If the security policy does require authentication, then reactive fragmentation MUST NOT be permitted if the partial bundle is not sufficient to allow authentication.

If reactive fragmentation is allowed, then all BAB blocks must be removed from created fragments.

5. Key Management

Key management in delay-tolerant networks is recognized as a difficult topic and is one that this specification does not attempt to solve.

6. Policy Considerations

When implementing the SBSP, 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.

- If a bundle is received that contains more than one security-operation, in violation of the SBSP, then the BPA must determine how to handle this 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 an intermediate receiver 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 SBSP blocks, replace the security-operation, or some other action.

- It is recommended that security operations only be applied to the payload block, the primary block, and any block-types specifically identified in the security policy. If a BPA were to apply security operations such as integrity or confidentiality to every block in the bundle, regardless of the block type, there could be downstream errors processing blocks whose contents must be inspected at every hop in the network path.

7. Security Considerations

Certain applications of DTN need to both sign and encrypt a message, and there are security issues to consider with this.

- To provide an assurance that a security-target came from a specific source and has not been changed, then it should be signed with a BIB.

- To ensure that a security-target cannot be inspected during transit, it should be encrypted with a BCB.

- Adding a BIB to a security-target that has already been encrypted by a BCB is not allowed. Therefore, we recommend three methods to add an integrity signature to an encrypted security-target. First, at the time of encryption, an integrity signature may be generated and added to the BCB for the security-target as additional information in the security-result field. Second, the encrypted block may be replicated as a new block and integrity signed. Third, 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.

8. Conformance

All implementations are strongly RECOMMENDED to provide at least a BAB cipher suite. A relay node, for example, might not deal with end-to-end confidentiality and data integrity, but it SHOULD exclude unauthorized traffic and perform hop-by-hop bundle verification.

9. IANA Considerations

This protocol has fields that have been registered by IANA.

9.1. Bundle Block Types

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

Additional Entries for the Bundle Block-Type Codes Registry:

Value	Description	Reference
2	Bundle Authentication Block	This document
3	Block Integrity Block	This document
4	Block Confidentiality Block	This document

Table 2

9.2. Cipher Suite Flags

This protocol has a cipher suite flags field and certain flags are defined. An IANA registry has been set up as follows.

The registration policy for this registry is: Specification Required

The Value range is: Variable Length

Cipher Suite Flag Registry:

Bit Position (right to left)	Description	Reference
0	Block contains result	This document
1	Block Contains parameters	This document
2	Source EID ref present	This document
>3	Reserved	This document

Table 3

9.3. Parameters and Results

This protocol has fields for cipher suite parameters and results. The field is a type-length-value triple and a registry is required for the "type" sub-field. The values for "type" apply to both the cipher suite parameters and the cipher suite results fields. Certain values are defined. An IANA registry has been set up as follows.

The registration policy for this registry is: Specification Required

The Value range is: 8-bit unsigned integer.

Cipher Suite Parameters and Results Type Registry:

Value	Description	Reference
0	reserved	This document
1	initialization vector (IV)	This document
2	reserved	This document
3	key-information	This document
4	content-range (pair of SDNVs)	This document
5	integrity signature	This document
6	unassigned	This document
7	salt	This document
8	BCB integrity check value (ICV)	This document
9-191	reserved	This document
192-250	private use	This document
251-255	reserved	This document

Table 4

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, November 2007.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

[RFC6255] Blanchet, M., "Delay-Tolerant Networking Bundle Protocol IANA Registries", RFC 6255, May 2011.

10.2. Informative References

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, April 2007.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", RFC 6257, May 2011.

Appendix A. Acknowledgements

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

Authors' Addresses

Edward J. Birrane, III
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
US

Phone: +1 443 778 7423
Email: Edward.Birrane@jhuapl.edu

Jeremy Pierce-Mayer
INSYEN AG
Muenchner Str. 20
Oberpfaffenhofen, Bavaria DE
Germany

Phone: +49 08153 28 2774
Email: jeremy.mayer@insyen.com

Dennis C. Iannicca
NASA Glenn Research Center
21000 Brookpark Rd.
Brook Park, OH 44135
US

Phone: +1-216-433-6493
Email: dennis.c.iannicca@nasa.gov