Delay-Tolerant Networking
Internet-Draft
Intended status: Standards Track
Expires: November 18, 2021

E. Birrane
A. White
S. Heiner
JHU/APL

May 17, 2021

BPSec Default Security Contexts
draft-ietf-dtn-bpsec-default-sc-07

Abstract

This document defines default integrity and confidentiality security contexts that can be used with the Bundle Protocol Security Protocol (BPSec) implementations. These security contexts are intended to be used for both testing the interoperability of BPSec implementations and for providing basic security operations when no other security contexts are defined or otherwise required for a network.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 18, 2021.

Table of Contents

1. Introduction

The Bundle Protocol Security Protocol (BPSec) [I-D.ietf-dtn-bpsec] specification provides inter-bundle integrity and confidentiality operations for networks deploying the Bundle Protocol (BP) [I-D.ietf-dtn-bpbis]. BPSec defines BP extension blocks to carry security information produced under the auspices of some security context.

This document defines two security contexts (one for an integrity service and one for a confidentiality service) for populating BPSec Block Integrity Blocks (BIBs) and Block Confidentiality Blocks (BCBs).

These contexts generate information that MUST be encoded using the CBOR specification documented in [RFC8949].

2. Requirements Language

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.

3. Integrity Security Context BIB-HMAC-SHA2

3.1. Overview

The BIB-HMAC-SHA2 security context provides a keyed hash over a set of plain text information. This context uses the Secure Hash Algorithm 2 (SHA-2) discussed in [SHS] combined with the HMAC keyed hash discussed in [HMAC]. The combination of HMAC and SHA-2 as the integrity mechanism for this security context was selected for two reasons:

1. The use of symmetric keys allows this security context to be used in places where an asymmetric-key infrastructure (such as a public key infrastructure) might be impractical.

2. The combination HMAC-SHA2 represents a well-supported and well-understood integrity mechanism with multiple implementations available.

BIB-HMAC-SHA2 supports three variants of HMAC-SHA, based on the supported length of the SHA-2 hash value. These variants correspond to "HMAC 256/256", "HMAC 384/384", and "HMAC 512/512" as defined in [RFC8152] Table 7: HMAC Algorithm Values. The selection of which variant is used by this context is provided as a security context parameter.

The output of the HMAC MUST be equal to the size of the SHA2 hashing function: 256 bits for SHA-256, 384 bits for SHA-384, and 512 bits for SHA-512.

The BIB-HMAC-SHA2 security context MUST have the security context identifier specified in Section 5.1.

3.2. Scope

The scope of BIB-HMAC-SHA2 is the set of information used to produce the plain text over which a keyed hash is calculated. This plain text is termed the "Integrity Protected Plain Text" (IPPT). The content of the IPPT is constructed as the concatenation of information whose integrity is being preserved from the BIB-HMAC-SHA2

security source to its security acceptor. There are four types of information that can be used in the generation of the IPPT, based on how broadly the concept of integrity is being applied. These four types of information, whether they are required, and why they are important for integrity, are discussed as follows.

Security target contents

The contents of the block-type-specific data field of the security target MUST be included in the IPPT. Including this information protects the security target data and is considered the minimal, required set of information for an integrity service on the security target.

Primary block

The primary block identifies a bundle and, once created, the contents of this block are immutable. Changes to the primary block associated with the security target indicate that the security target (and BIB) might no longer be in the correct bundle.

For example, if a security target and associated BIB are copied from one bundle to another bundle, the BIB might still contain a verifiable signature for the security target unless information associated with the bundle primary block is included in the keyed hash carried by the BIB.

Including this information in the IPPT protects the integrity of the association of the security target with a specific bundle.

Security target other fields

The other fields of the security target include block identification and processing information. Changing this information changes how the security target is treated by nodes in the network even when the "user data" of the security target are otherwise unchanged.

For example, if the block processing control flags of a security target are different at a security verifier than they were originally set at the security source then the policy for handling the security target has been modified.

Including this information in the IPPT protects the integrity of the policy and identification of the security target data.

BIB other fields

The other fields of the BIB include block identification and processing information. Changing this information changes how

the BIB is treated by nodes in the network, even when other aspects of the BIB are unchanged.

For example, if the block processing control flags of the BIB are different at a security verifier than they were originally set at the security source, then the policy for handling the BIB has been modified.

Including this information in the IPPT protects the integrity of the policy and identification of the security service in the bundle.

NOTE: The security context identifier and security context parameters of the security block are not included in the IPPT because these parameters, by definition, are required to verify or accept the security service. Successful verification at security verifiers and security acceptors implies that these parameters were unchanged since being specified at the security source.

The scope of the BIB-HMAC-SHA2 security context is configured using an optional security context parameter.

3.3. Parameters

BIB-HMAC-SHA2 can be parameterized to select SHA-2 variants, communicate key information, and define the scope of the IPPT.

3.3.1. SHA Variant

This optional parameter identifies which variant of the SHA-2 algorithm is to be used in the generation of the authentication code.

This value MUST be encoded as a CBOR unsigned integer.

Valid values for this parameter are as follows.

SHA Variant Parameter Values

Value	Description
5	HMAC 256/256 as defined in [RFC8152] Table 7: HMAC Algorithm Values
6	HMAC 384/384 as defined in [RFC8152] Table 7: HMAC Algorithm Values
7	HMAC 512/512 as defined in [RFC8152] Table 7: HMAC Algorithm Values

Table 1

When not provided, implementations SHOULD assume a value of 6 (indicating use of HMAC 384/384), unless an alternate default is established by local security policy at the security source, verifiers, or acceptor of this integrity service.

3.3.2. Wrapped Key

This optional parameter contains the output of the AES key wrap authenticated encryption function (KW-AE) as defined in [AES-KW]. Specifically, this parameter holds the cipher text produced when running the KW-AE algorithm with the input string being the symmetric HMAC key used to generate the security results present in the security block. The value of this parameter is used as input to the AES key wrap authenticated decryption function (KW-AD) at security verifiers and security acceptors to determine the symmetric HMAC key needed for the proper validation of the security results in the security block.

This value MUST be encoded as a CBOR byte string.

If this parameter is not present then security verifiers and acceptors MUST determine the proper key as a function of their local BPSec policy and configuration.

3.3.3. Integrity Scope Flags

This optional parameter contains a series of flags that describe what information is to be included with the block-type-specific data when constructing the IPPT value.

This value MUST be represented as a CBOR unsigned integer, the value of which MUST be processed as a bit field.

Integrity scope flags that are unrecognized MUST be ignored, as future definitions of additional flags might not be integrated simultaneously into security context implementations operating at all nodes.

Bits in this field represent additional information to be included when generating an integrity signature over the security target. These bits are defined as follows.

- Bit 0 (the low-order bit, 0x0001): Primary Block Flag.

- Bit 1 (0x0002): Target Header Flag.

- Bit 2 (0x0003): Security Header Flag.

- Bits 3-7 are reserved.

- Bits 8-15 are unassigned.

3.3.4. Enumerations

BIB-HMAC-SHA2 defines the following security context parameters.

BIB-HMAC-SHA2 Security Parameters

Id	Name	CBOR Encoding Type	Default Value
1	SHA Variant	UINT	6
2	Wrapped Key	Byte String	NONE
4	Integrity Scope Flags	UINT	0x7

Table 2

3.4. Results

BIB-HMAC-SHA2 defines the following security results.

BIB-HMAC-SHA2 Security Results

Result Id	Result Name	CBOR Encoding Type	Description
1	Expected HMAC	byte string	The output of the HMAC calculation at the security source.

Table 3

3.5. Key Considerations

HMAC keys used with this context MUST be symmetric and MUST have a key length equal to the output of the HMAC. For this reason, HMAC keys will be integer divisible by 8 bytes and special padding-aware AES key wrap algorithms are not needed.

It is assumed that any security verifier or security acceptor performing an integrity verification can determine the proper HMAC key to be used. Potential sources of the HMAC key include (but are not limited to) the following:

Pre-placed keys selected based on local policy.

Keys extracted from material carried in the BIB.

Session keys negotiated via a mechanism external to the BIB.

When an AES-KW wrapped key is present in a security block, it is assumed that security verifiers and security acceptors can independently determine the key encryption key (KEK) used in the wrapping of the symmetric HMAC key.

As discussed in Section 6 and emphasized here, it is strongly recommended that keys be protected once generated, both when they are stored and when they are transmitted.

3.6. Canonicalization Algorithms

This section defines the canonicalization algorithm used to prepare the IPPT input to the BIB-HMAC-SHA2 integrity mechanism. The construction of the IPPT depends on the settings of the integrity scope flags that can be provided as part of customizing the behavior of this security context.

In all cases, the canonical form of any portion of an extension block MUST be performed as described in [I-D.ietf-dtn-bpsec]. The canonicalization algorithms defined in [I-D.ietf-dtn-bpsec] adhere to the canonical forms for extension blocks defined in [I-D.ietf-dtn-bpbis] but resolve ambiguities related to how values are represented in CBOR.

The IPPT is constructed using the following process.

1. The canonical form of the IPPT starts as the empty set with length 0.

2. If the integrity scope parameter is present and the primary block flag is set to 1, then a canonical form of the bundle's primary block MUST be calculated and the result appended to the IPPT.

3. If the integrity scope parameter is present and the target header flag is set to 1, then the canonical form of the block type code, block number, and block processing control flags associated with the security target MUST be calculated and, in that order, appended to the IPPT.

4. If the integrity scope parameter is present and the security header flag is set to 1, then the canonical form of the block type code, block number, and block processing control flags associated with the BIB MUST be calculated and, in that order, appended to the IPPT.

5. The canonical form of the security target block-type-specific data MUST be calculated and appended to the IPPT.

3.7. Processing

3.7.1. Keyed Hash Generation

During keyed hash generation, two inputs are prepared for the the appropriate HMAC/SHA2 algorithm: the HMAC key and the IPPT. These data items MUST be generated as follows.

The HMAC key MUST have the appropriate length as required by local security policy. The key can be generated specifically for this integrity service, given as part of local security policy, or through some other key management mechanism as discussed in Section 3.5.

Prior to the generation of the IPPT, if a CRC value is present for the target block of the BIB, then that CRC value MUST be removed from the target block. This involves both removing the CRC value

from the target block and setting the CRC Type field of the target block to "no CRC is present."

Once CRC information is removed, the IPPT MUST be generated as discussed in Section 3.6.

Upon successful hash generation the following actions MUST occur.

The keyed hash produced by the HMAC/SHA2 variant MUST be added as a security result for the BIB representing the security operation on this security target, as discussed in Section 3.4).

Finally, the BIB containing information about this security operation MUST be updated as follows. These operations can occur in any order.

The security context identifier for the BIB MUST be set to the context identifier for BIB-HMAC-SHA2.

Any local flags used to generate the IPPT SHOULD be placed in the integrity scope flags security parameter for the BIB unless these flags are expected to be correctly configured at security verifiers and acceptors in the network.

The HMAC key MAY be wrapped using the NIST AES-KW algorithm and the results of the wrapping added as the wrapped key security parameter for the BIB.

The SHA variant used by this security context SHOULD be added as the SHA variant security parameter for the BIB if it differs from the default key length. Otherwise, this parameter MAY be omitted if doing so provides a useful reduction in message sizes.

Problems encountered in the keyed hash generation MUST be processed in accordance with local BPSec security policy.

3.7.2. Keyed Hash Verification

During keyed hash verification, the input of the security target and a HMAC key are provided to the appropriate HMAC/SHA2 algorithm.

During keyed hash verification, two inputs are prepared for the appropriate HMAC/SHA2 algorithm: the HMAC key and the IPPT. These data items MUST be generated as follows.

The HMAC key MUST be derived using the wrapped key security parameter if such a parameter is included in the security context parameters of the BIB. Otherwise, this key MUST be derived in

accordance with security policy at the verifying node as discussed in Section 3.5.

The IPPT MUST be generated as discussed in Section 3.6 with the value of integrity scope flags being taken from the integrity scope flags security context parameter. If the integrity scope flags parameter is not included in the security context parameters then these flags MAY be derived from local security policy.

The calculated HMAC output MUST be compared to the expected HMAC output encoded in the security results of the BIB for the security target. If the calculated HMAC and expected HMAC are identical, the verification MUST be considered a success. Otherwise, the verification MUST be considered a failure.

If the verification fails or otherwise experiences an error, or if any needed parameters are missing, then the verification MUST be treated as failed and processed in accordance with local security policy.

This security service is removed from the bundle at the security acceptor as required by the BPSec specification. If the security acceptor is not the bundle destination and if no other integrity service is being applied to the target block, then a CRC MUST be included for the target block. The CRC type, as determined by policy, is set in the target block's CRC type field and the corresponding CRC value is added as the CRC field for that block.

4. Security Context BCB-AES-GCM

4.1. Overview

The BCB-AES-GCM security context replaces the block-type-specific data field of its security target with cipher text generated using the Advanced Encryption Standard (AES) cipher operating in Galois/Counter Mode (GCM) [AES-GCM]. The use of AES-GCM was selected as the cipher suite for this confidentiality mechanism for several reasons:

1. The selection of a symmetric-key cipher suite allows for relatively smaller keys than asymmetric-key cipher suites.

2. The selection of a symmetric-key cipher suite allows this security context to be used in places where an asymmetric-key infrastructure (such as a public key infrastructure) might be impractical.

3. The use of the Galois/Counter Mode produces cipher-text with the same size as the plain text making the replacement of target

block information easier as length fields do not need to be changed.

4. The AES-GCM cipher suite provides authenticated encryption, as required by the BPSec protocol.

Additionally, the BCB-AES-GCM security context generates an authentication tag based on the plain text value of the block-type-specific data and other additional authenticated data that might be specified via parameters to this security context.

This security context supports two variants of AES-GCM, based on the supported length of the symmetric key. These variants correspond to A128GCM and A256GCM as defined in [RFC8152] Table 9: Algorithm Value for AES-GCM.

The BCB-AES-GCM security context MUST have the security context identifier specified in Section 5.1.

4.2. Scope

There are two scopes associated with BCB-AES-GCM: the scope of the confidentiality service and the scope of the authentication service. The first defines the set of information provided to the AES-GCM cipher for the purpose of producing cipher text. The second defines the set of information used to generate an authentication tag.

The scope of the confidentiality service defines the set of information provided to the AES-GCM cipher for the purpose of producing cipher text. This MUST be the full set of plain text contained in the block-type-specific data field of the security target.

The scope of the authentication service defines the set of information used to generate an authentication tag carried with the security block. This information includes the data included in the confidentiality service and MAY include other information (additional authenticated data), as follows.

Primary block
: The primary block identifies a bundle and, once created, the contents of this block are immutable. Changes to the primary block associated with the security target indicate that the security target (and BCB) might no longer be in the correct bundle.

  For example, if a security target and associated BCB are copied from one bundle to another bundle, the BCB might still be able to

decrypt the security target even though these blocks were never intended to exist in the copied-to bundle.

Including this information as part of additional authenticated data ensures that security target (and security block) appear in the same bundle at the time of decryption as at the time of encryption.

Security target other fields

The other fields of the security target include block identification and processing information. Changing this information changes how the security target is treated by nodes in the network even when the "user data" of the security target are otherwise unchanged.

For example, if the block processing control flags of a security target are different at a security verifier than they were originally set at the security source then the policy for handling the security target has been modified.

Including this information as part of additional authenticated data ensures that the cipher text in the security target will not be used with a different set of block policy than originally set at the time of encryption.

BCB other fields

The other fields of the BCB include block identification and processing information. Changing this information changes how the BCB is treated by nodes in the network, even when other aspects of the BCB are unchanged.

For example, if the block processing control flags of the BCB are different at a security acceptor than they were originally set at the security source then the policy for handling the BCB has been modified.

Including this information as part of additional authenticated data ensures that the policy and identification of the security service in the bundle has not changed.

NOTE: The security context identifier and security context parameters of the security block are not included as additional authenticated data because these parameters, by definition, are those needed to verify or accept the security service. Therefore, it is expected that changes to these values would result in failures at security verifiers and security acceptors.

The scope of the BCB-AES-GCM security context is configured using an optional security context parameter.

4.3. Parameters

BCB-AES-GCM can be parameterized to specify the AES variant, initialization vector, key information, and identify additional authenticated data.

4.3.1. Initialization Vector (IV)

This optional parameter identifies the initialization vector (IV) used to initialize the AES-GCM cipher.

The length of the initialization vector, prior to any CBOR encoding, MUST be between 8-16 bytes. A value of 12 bytes SHOULD be used unless local security policy requires a different length.

This value MUST be encoded as a CBOR byte string.

The initialization vector can have any value with the caveat that a value MUST NOT be re-used for multiple encryptions using the same encryption key. This value MAY be re-used when encrypting with different keys. For example, if each encryption operation using BCB-AES-GCM uses a newly generated key, then the same IV can be reused.

4.3.2. AES Variant

This optional parameter identifies the AES variant being used for the AES-GCM encryption, where the variant is identified by the length of key used.

This value MUST be encoded as a CBOR unsigned integer.

Valid values for this parameter are as follows.

AES Variant Parameter Values

Value	Description
1	A128GCM as defined in [RFC8152] Table 9: Algorithm Values for AES-GCM
3	A256GCM as defined in [RFC8152] Table 9: Algorithm Values for AES-GCM

When not provided, implementations SHOULD assume a value of 3 (indicating use of A256GCM), unless an alternate default is established by local security policy at the security source, verifier, or acceptor of this integrity service.

Regardless of the variant, the generated authentication tag MUST always be 128 bits.

4.3.3. Wrapped Key

This optional parameter contains the output of the AES key wrap authenticated encryption function (KW-AE) as defined in [AES-KW]. Specifically, this parameter holds the cipher text produced when running the KW-AE algorithm with the input string being the symmetric AES key used to generate the security results present in the security block. The value of this parameter is used as input to the AES key wrap authenticated decryption function (KW-AD) at security verifiers and security acceptors to determine the symmetric AES key needed for the proper decryption of the security results in the security block.

This value MUST be encoded as a CBOR byte string.

If this parameter is not present then security verifiers and acceptors MUST determine the proper key as a function of their local BPSec policy and configuration.

4.3.4. AAD Scope Flags

This optional parameter contains a series of flags that describe what information is to be included with the block-type-specific data of the security target as part of additional authenticated data (AAD).

This value MUST be represented as a CBOR unsigned integer, the value of which MUST be processed as a bit field.

AAD scope flags that are unrecognized MUST be ignored, as future definitions of additional flags might not be integrated simultaneously into security context implementations operating at all nodes.

Bits in this field represent additional information to be included when generating an integrity signature over the security target. These bits are defined as follows.

- Bit 0 (the low-order bit, 0x0001): Primary Block Flag.

- Bit 1 (0x0002): Target Header Flag.

- Bit 2 (0x0003): Security Header Flag.

- Bits 3-7 are reserved.

- Bits 8-15 are unassigned.

4.3.5. Enumerations

BCB-AES-GCM defines the following security context parameters.

BCB-AES-GCM Security Parameters

Id	Name	CBOR Encoding Type	Default Value
1	Initialization Vector	Byte String	NONE
2	AES Variant	UINT	3
3	Wrapped Key	Byte String	NONE
4	AAD Scope Flags	UINT	0x7

Table 4

4.4. Results

The BCB-AES-GCM security context produces a single security result carried in the security block: the authentication tag.

NOTES:

The cipher text generated by the cipher suite is not considered a security result as it is stored in the block-type-specific data field of the security target block. When operating in GCM mode, AES produces cipher text of the same size as its plain text and, therefore, no additional logic is required to handle padding or overflow caused by the encryption in most cases (see below).

If the generated cipher text contains the authentication tag and the tag can be separated from the cipher text then the tag MUST be separated and stored in the authentication tag security result field.

If the generated cipher text contains the authentication tag and the tag cannot be separated from the cipher text then the tag MUST NOT be included in the authentication tag security result field. Instead the security target block MUST be resized to accommodate the additional 128 bits of authentication tag included in the generated cipher text.

4.4.1. Authentication Tag

The authentication tag is generated by the cipher suite over the security target plain text input to the cipher suite as combined with any optional additional authenticated data. This tag is used to ensure that the plain text (and important information associated with the plain text) is authenticated prior to decryption.

If the authentication tag is included in the cipher text placed in the security target block-type-specific data field, then this security result MUST NOT be included in the BCB for that security target.

The length of the authentication tag, prior to any CBOR encoding, MUST be 128 bits.

This value MUST be encoded as a CBOR byte string.

4.4.2. Enumerations

BCB-AES-GCM defines the following security context parameters.

BCB-AES-GCM Security Results

Result Id	Result Name	CBOR Encoding Type
1	Authentication Tag	Byte String

Table 5

4.5. Key Considerations

Keys used with this context MUST be symmetric and MUST have a key length equal to the key length defined in the security context parameters or as defined by local security policy at security verifiers and acceptors. For this reason, content-encrypting keys will be integer divisible by 8 bytes and special padding-aware AES key wrap algorithms are not needed.

It is assumed that any security verifier or security acceptor can determine the proper key to be used. Potential sources of the key include (but are not limited to) the following.

Pre-placed keys selected based on local policy.

Keys extracted from material carried in the BCB.

Session keys negotiated via a mechanism external to the BCB.

When an AES-KW wrapped key is present in a security block, it is assumed that security verifiers and security acceptors can independently determine the key encryption key (KEK) used in the wrapping of the symmetric AES content-encrypting key.

The security provided by block ciphers is reduced as more data is processed with the same key. The total number of bytes processed with a single key for AES-GCM is recommended to be less than 2^64, as described in Appendix B of [AES-GCM].

As discussed in Section 6 and emphasized here, it is strongly recommended that keys be protected once generated, both when they are stored and when they are transmitted.

4.6. GCM Considerations

The GCM cryptographic mode of AES has specific requirements that MUST be followed by implementers for the secure function of the BCB-AES-GCM security context. While these requirements are well documented in [AES-GCM], some of them are repeated here for emphasis.

The pairing of an IV and a security key MUST be unique. An IV MUST NOT be used with a security key more than one time. If an IV and key pair are repeated then the GCM implementation is vulnerable to forgery attacks. More information regarding the importance of the uniqueness of the IV value can be found in Appendix A of [AES-GCM].

While any tag-based authentication mechanism has some likelihood of being forged, this probability is increased when using AES-GCM. In particular, short tag lengths combined with very long messages SHOULD be avoided when using this mode. The BCB-AES-GCM security context requires the use of 128-bit authentication tags at all times. Concerns relating to the size of authentication tags is discussed in Appendices B and C of [AES-GCM].

As discussed in Appendix B of [AES-GCM], implementations SHOULD limit the number of unsuccessful verification attempts for each key to reduce the likelihood of guessing tag values.

As discussed in the Security Considerations section of [I-D.ietf-dtn-bpsec], delay-tolerant networks have a higher occurrence of replay attacks due to the store-and-forward nature of the network. Because GCM has no inherent replay attack protection, implementors SHOULD attempt to detect replay attacks

by using mechanisms such as those described in Appendix D of [AES-GCM].

4.7. Canonicalization Algorithms

This section defines the canonicalization algorithms used to prepare the inputs used to generate both the cipher text and the authentication tag.

In all cases, the canonical form of any portion of an extension block MUST be performed as described in [I-D.ietf-dtn-bpsec]. The canonicalization algorithms defined in [I-D.ietf-dtn-bpsec] adhere to the canonical forms for extension blocks defined in [I-D.ietf-dtn-bpbis] but resolve ambiguities related to how values are represented in CBOR.

4.7.1. Cipher text related calculations

The plain text used during encryption MUST be calculated as the single, definite-length CBOR byte string representing the block-type-specific data field of the security target excluding the CBOR byte string identifying byte and optional CBOR byte string length field.

For example, consider the following two CBOR byte strings and the plain text that would be extracted from them.

CBOR Byte String Examples

CBOR Byte String (Hex)	CBOR Part (Hex)	Plain Text Part (Hex)
18ED	18	ED
C24CDEADBEEFDEADBEEFDEADBEEF	C24C	DEADBEEFDEADBEEFDEADBEEF

Table 6

Similarly, the cipher text used during decryption MUST be calculated as the single, definite-length CBOR byte string representing the block-type-specific data field excluding the CBOR byte string identifying byte and optional CBOR byte string length field.

All other fields of the security target (such as the block type code, block number, block processing control flags, or any CRC information) MUST NOT be considered as part of encryption or decryption.

4.7.2. Additional Authenticated Data

The construction of additional authenticated data depends on the AAD scope flags that can be provided as part of customizing the behavior of this security context.

The canonical form of the AAD input to the BCB-AES-GCM mechanism is constructed using the following process. This process MUST be followed when generating AAD for either encryption or decryption.

1. The canonical form of the AAD starts as the empty set with length 0.

2. If the AAD scope parameter is present and the primary block flag is set to 1, then a canonical form of the bundle's primary block MUST be calculated and the result appended to the AAD.

3. If the AAD scope parameter is present and the target header flag is set to 1, then the canonical form of the block type code, block number, and block processing control flags associated with the security target MUST be calculated and, in that order, appended to the AAD.

4. If the AAD scope parameter is present and the security header flag is set to 1, then the canonical form of the block type code, block number, and block processing control flags associated with the BIB MUST be calculated and, in that order, appended to the AAD.

If, after this process, the AAD remains at length 0, then no AAD exists to be input to the cipher suite.

4.8. Processing

4.8.1. Encryption

During encryption, four inputs are prepared for input to the AES/GCM cipher: the encryption key, the IV, the security target plain text to be encrypted, and any additional authenticated data. These data items MUST be generated as follows.

Prior to encryption, if a CRC value is present for the target block, then that CRC value MUST be removed. This requires removing the CRC field from the target block and setting the CRC type field of the target block to "no CRC is present."

The encryption key MUST have the appropriate length as required by local security policy. The key might be generated specifically

for this encryption, given as part of local security policy, or through some other key management mechanism as discussed in Section 4.5.

The IV selected MUST be of the appropriate length. Because replaying an IV in counter mode voids the confidentiality of all messages encrypted with said IV, this context also requires a unique IV for every encryption performed with the same key. This means the same key and IV combination MUST NOT be used more than once.

The security target plain text for encryption MUST be generated as discussed in Section 4.7.1.

Additional authenticated data, if present, MUST be generated as discussed in Section 4.7.2 with the value of AAD scope flags being taken from local security policy.

Upon successful encryption the following actions MUST occur.

The cipher text produced by AES/GCM MUST replace the bytes used to define the plain text in the security target block's block-type-specific data field. The block length of the security target MUST be updated if the generated cipher text is larger than the plain text (which can occur when the authentication tag is included in the cipher text calculation, as discussed in Section 4.4).

The authentication tag calculated by the AES/GCM cipher MUST be added as a security result for the security target in the BCB holding results for this security operation.

Cases where the authentication tag is generated as part of the cipher text MUST be processed as described in Section 4.4.

Finally, the BCB containing information about this security operation MUST be updated as follows. These operations can occur in any order.

The security context identifier for the BCB MUST be set to the context identifier for BCB-AES-GCM.

The IV input to the cipher MUST be added as the IV security parameter for the BCB.

Any local flags used to generated AAD for this cipher MUST be added as the AAD scope flags security parameter for the BCB.

The encryption key MAY be wrapped using the NIST AES-KW algorithm and the results of the wrapping added as the wrapped key security parameter for the BCB.

The key length used by this security context MUST be considered when setting the AES variant security parameter for the BCB if it differs from the default AES variant. Otherwise, the AES variant MAY be omitted if doing so provides a useful reduction in message sizes.

Problems encountered in the encryption MUST be processed in accordance with local security policy. This MAY include restoring a CRC value removed from the target block prior to encryption, if the target block is allowed to be transmitted after an encryption error.

4.8.2. Decryption

During encryption, five inputs are prepared for input to the AES/GCM cipher: the decryption key, the IV, the security target cipher text to be decrypted, any additional authenticated data, and the authentication tag generated from the original encryption. These data items MUST be generated as follows.

The decryption key MUST be derived using the wrapped key security parameter if such a parameter is included in the security context parameters of the BCB. Otherwise this key MUST be derived in accordance with local security policy at the decrypting node as discussed in Section 4.5.

The IV MUST be set to the value of the IV security parameter included in the BCB. If the IV parameter is not included as a security parameter, an IV MAY be derived as a function of local security policy and other BCB contents or a lack of an IV security parameter in the BCB MAY be treated as an error by the decrypting node.

The security target cipher text for decryption MUST be generated as discussed in Section 4.7.1.

Additional authenticated data, if present, MUST be generated as discussed in Section 4.7.2 with the value of AAD scope flags being taken from the AAD scope flags security context parameter. If the AAD scope flags parameter is not included in the security context parameters then these flags MAY be derived from local security policy in cases where the set of such flags is determinable in the network.

The authentication tag MUST be present in the BCB security context parameters field if additional authenticated data are defined for the BCB (either in the AAD scope flags parameter or as specified by local policy). This tag MUST be 128 bits in length.

Upon successful decryption the following actions MUST occur.

The plain text produced by AES/GCM MUST replace the bytes used to define the cipher text in the security target block's block-type-specific data field. Any changes to the security target block length field MUST be corrected in cases where the plain text has a different length than the replaced cipher text.

If the security acceptor is not the bundle destination and if no other integrity or confidentiality service is being applied to the target block, then a CRC MUST be included for the target block. The CRC type, as determined by policy, is set in the target block's CRC type field and the corresponding CRC value is added as the CRC field for that block.

If the cipher text fails to authenticate, if any needed parameters are missing, or if there are other problems in the decryption then the decryption MUST be treated as failed and processed in accordance with local security policy.

5. IANA Considerations

5.1. Security Context Identifiers

This specification allocates two security context identifiers from the "BPSec Security Context Identifier" registry defined in [I-D.ietf-dtn-bpsec].

Additional Entries for the BPSec Security Context Identifiers Registry:

Value	Description	Reference
TBA	BIB-HMAC-SHA2	This document
TBA	BCB-AES-GCM	This document

Table 7

5.2. Integrity Scope Flags

The BIB-HMAC-SHA2 security context has an Integrity Scope Flags field for which IANA is requested to create and maintain a new registry named "BPSec BIB-HMAC-SHA2 Integrity Scope Flags." Initial values for this registry are given below.

The registration policy for this registry is: Specification Required.

The value range is unsigned 16-bit integer.

BPSec BIB-HMAC-SHA2 Integrity Scope Flags Registry

Bit Position (right to left)	Description	Reference
0	Include primary block	This document
1	Include target header flag	This document
2	Include security header flag	This document
3-7	reserved	This document
8-15	unassigned	This document

Table 8

5.3. AAD Scope Flags

The BCB-AES-GCM security context has an AAD Scope Flags field for which IANA is requested to create and maintain a new registry named "BPSec BCB-AES-GCM AAD Scope Flags." Initial values for this registry are given below.

The registration policy for this registry is: Specification Required.

The value range is unsigned 16-bit integer.

BPSec BCB-AES-GCM AAD Scope Flags Registry

Bit Position (right to left)	Description	Reference
0	Include primary block	This document
1	Include target header flag	This document
2	Include security header flag	This document
3-7	reserved	This document
8-15	unassigned	This document

Table 9

6. Security Considerations

Security considerations specific to a single security context are provided in the description of that context. This section discusses security considerations that should be evaluated by implementers of any security context described in this document. Considerations can also be found in documents listed as normative references and they should also be reviewed by security context implementors.

6.1. Key Management

The delayed and disrupted nature of DTNs complicates the process of key management because there might not be reliable, timely round-trip exchange between security sources, security verifiers, and security acceptors in the network. This is true when there is a substantial signal propagation delay between nodes, when nodes are in a highly challenged communications environment, and when nodes do not support bi-directional communication.

In these environments, key establishment protocols that rely on round-trip information exchange might not converge on a shared secret in a timely manner (or at all). Also, key revocation or key verification mechanisms that rely on access to a centralized authority (such as a certificate authority) might similarly fail in the stressing conditions of a DTN.

For these reasons, the default security contexts described in this document rely on symmetric key cryptographic mechanisms because

asymmetric key infrastructure (such as a public key infrastructure) is impractical in this environment. This extends to any asymmetric-key mechanism for key derivation, key exchange, or key revocation.

BPSec assumes that "key management is handled as a separate part of network management" [I-D.ietf-dtn-bpsec]. This assumption is also made by the security contexts defined in this document which do not define new protocols for key derivation, exchange of key-encrypting keys, revocation of existing keys, or the security configuration or policy used to select certain keys for certain security operations.

Nodes using these security contexts need to perform the following kinds of activities, independent of the construction, transmission, and processing of BPSec security blocks.

Establish shared key-encrypting-keys with other nodes in the network using an out-of-band mechanism. This might include pre-sharing of key encryption keys or the use of traditional key establishment mechanisms prior to the exchange of BPsec security blocks.

Determine when a key is considered exhausted and no longer to be used in the generation, verification, or acceptance of a security block.

Determine when a key is considered invalid and no longer to be used in the generation, verification, or acceptance of a security block. Such revocations can be based on a variety of mechanisms to include local security policy, time relative to the generation or use of the key, or as specified through network management.

Determine, through an out-of-band mechanism such as local security policy, what keys are to be used for what security blocks. This includes the selection of which key should be used in the evaluation of a security block received by a security verifier or a security acceptor.

The failure to provide effective key management techniques appropriate for the operational networking environment can result in the compromise of those unmanaged keys and the loss of security services in the network.

6.2. Key Handling

Once generated, keys should be handled as follows.

It is strongly RECOMMENDED that implementations protect keys both when they are stored and when they are transmitted.

In the event that a key is compromised, any security operations using a security context associated with that key SHOULD also be considered compromised. This means that the BIB-HMAC-SHA2 security context SHOULD NOT provide integrity when used with a compromised key and BCB-AES-GCM SHOULD NOT provide confidentiality when used with a compromised key.

The same key SHOULD NOT be used for different algorithms as doing so might leak information about the key.

6.3. AES GCM

There are a significant number of considerations related to the use of the GCM mode of AES to provide a confidentiality service. These considerations are provided in Section 4.6 as part of the documentation of the BCB-AES-GCM security context.

It should be noted that the authentication tag produced by the GCM mode of AES is not considered part of the cipher text itself. While certain implementations might catenate the authentication tag and the produced cipher text, they are distinct values. For this reason, the authentication tag is expected to be carried in the BCB-AES-GCM security block as a security result whenever the authentication tag can be differentiated from the produced cipher text.

6.4. AES Key Wrap

The AES key wrap (AES-KW) algorithm used by the security contexts in this document does not use an initialization vector and does not require any key padding. Key padding is not needed because wrapped keys used by these security contexts will always be multiples of 8 bytes. The length of the wrapped key can be determined by inspecting the security context parameters. Therefore, a key can be unwrapped using only the information present in the security block and the key encryption key provided by local security policy at the security verifier or security acceptor.

6.5. Bundle Fragmentation

Bundle fragmentation might prevent security services in a bundle from being verified after a bundle is fragmented and before the bundle is re-assembled. Examples of potential issues include the following.

If a security block and its security target do not exist in the same fragment, then the security block cannot be processed until the bundle is re-assembled. If a fragment includes an encrypted target block, but not its BCB, then a receiving bundle processing

agent (BPA) will not know that the target block has been encrypted.

If a security block is cryptographically bound to a bundle, it cannot be processed even if the security block and target both coexist in the fragment. This is because fragments have different primary blocks than the original bundle.

If security blocks and their target blocks are repeated in multiple fragments, policy needs to determine how to deal with issues where a security operation verifies in one fragment but fails in another fragment. This might happen, for example, if a BIB block becomes corrupted in one fragment but not in another fragment.

Implementors should consider how security blocks are processed when a BPA fragments a received bundle. For example, security blocks and their targets could be placed in the same fragment if the security block is not otherwise cryptographically bound to the bundle being fragmented. Alternatively, if security blocks are cryptographically bound to a bundle, then a fragmenting BPA should consider encapsulating the bundle first and then fragmenting the encapsulating bundle.

7. Normative References

[AES-GCM] Dworkin, M., "NIST Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.", November 2007.

[AES-KW] Dworkin, M., "NIST Special Publication 800-38F: Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping.", December 2012.

[HMAC] US NIST, "The Keyed-Hash Message Authentication Code (HMAC).", FIPS-198-1, Gaithersburg, MD, USA, July 2008.

https://csrc.nist.gov/publications/detail/fips/198/1/final

[I-D.ietf-dtn-bpbis] Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol Version 7", draft-ietf-dtn-bpbis-31 (work in progress), January 2021.

[I-D.ietf-dtn-bpsec] Birrane, E. and K. McKeever, "Bundle Protocol Security Specification", draft-ietf-dtn-bpsec-27 (work in progress), February 2021.

[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>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[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>.

[SHS] US NIST, "Secure Hash Standard (SHS).", FIPS-180-4, Gaithersburg, MD, USA, August 2015.

https://csrc.nist.gov/publications/detail/fips/180/4/final

Appendix A. Examples

This appendix is informative.

This section presents a series of examples of constructing BPSec security blocks (using the security contexts defined in this document) and adding those blocks to a sample bundle.

The examples presented in this appendix represent valid constructions of bundles, security blocks, and the encoding of security context parameters and results. For this reason, they can inform unit test suites for individual implementations as well as interoperability test suites amongst implementations. However, these examples do not cover every permutation of security parameters, security results, or use of security blocks in a bundle.

NOTE: Figures in this section identified as "(CBOR Diagnostic Notation)" are represented using the CBOR diagnostic notation defined in [RFC8949]. This notation is used to express CBOR data structures in a manner that enables visual inspection. The bundles, security blocks, and security context contents in these figures are represented using CBOR structures.

NOTE: Examples in this section use the "ipn" URI scheme for EndpointID naming, as defined in [I-D.ietf-dtn-bpbis].

NOTE: The bundle source is presumed to be the security source for all security blocks in this section, unless otherwise noted.

A.1. Example 1: Simple Integrity

This example shows the addition of a BIB to a sample bundle to provide integrity for the payload block.

A.1.1. Original Bundle

The following diagram shows the original bundle before the BIB has been added.

```
                      Block                    Block   Block
                    in Bundle                  Type    Number
     +=========================================+=======+========+
     |  Primary Block                          |  N/A  |    0   |
     +-----------------------------------------+-------+--------+
     |  Payload Block                          |   0   |    1   |
     +-----------------------------------------+-------+--------+
```

Figure 1: Example 1 Original Bundle

A.1.1.1. Primary Block

The BPv7 bundle has no special processing flags and no CRC is provided because the primary block is expected to be protected by an integrity service BIB using the BIB-HMAC-SHA2 security context.

The bundle is sourced at the source node ipn:2.1 and destined for the destination node ipn:1.2. The bundle creation time uses a DTN creation time of 0 indicating lack of an accurate clock and a sequence number of 40. The lifetime of the bundle is given as 1,000,000 milliseconds since the bundle creation time.

The primary block is provided as follows.

```
[
  7,           / BP version            /
  0,           / flags                 /
  0,           / CRC type              /
  [2, [1,2]],  / destination (ipn:1.2) /
  [2, [2,1]],  / source      (ipn:2.1) /
  [2, [2,1]],  / report-to   (ipn:2.1) /
  [0, 40],     / timestamp             /
  1000000      / lifetime              /
]
```

Figure 2: Primary Block (CBOR Diagnostic Notation)

The CBOR encoding of the primary block is
0x88070000820282010282028202018202820201820018281a000f4240.

A.1.1.2. Payload Block

Other than its use as a source of plaintext for security blocks, the payload has no required distinguishing characteristic for the purpose of this example. The sample payload is a 32 byte string whose value is "Ready Generate a 32 byte payload".

The payload is represented in the payload block as a byte string of the raw payload string. It is NOT represented as a CBOR text string wrapped within a CBOR binary string. The hex value of the payload "Ready Generate a 32 byte payload" is
0x52656164792047656e65726174652061203332206279746520706179 6c6f6164.

The payload block is provided as follows.

```
[
  1,      / type code: Payload block /
  1,      / block number             /
  0,      / block processing flags   /
  0,      / CRC Type                 /
  h'52656164792047656e65726174652061 / type-specific-data: payload /
    203332206279746520706179 6c6f6164'

]
```

Payload Block (CBOR Diagnostic Notation)

The CBOR encoding of the payload block is 0x850101000058205265616479 2047656e65726174652061203332206279746520706179 6c6f6164.

A.1.1.3. Bundle CBOR Representation

A BPv7 bundle is represented as an indefinite-length array consisting of the blocks comprising the bundle, with a terminator character at the end.

The CBOR encoding of the original bundle is 0x9f88070000820282010282028202018202820201820018281a000f4240850101000058205265616479204765 6e657261746520612033322062797465207061796c6f6164ff.

A.1.2. Security Operation Overview

This example adds a BIB to the bundle using the BIB-HMAC-SHA2 security context to provide an integrity mechanism over the payload block.

The following diagram shows the resulting bundle after the BIB is added.

```
                      Block                    Block   Block
                    in Bundle                  Type    Number
     +=========================================+=======+========+
     |  Primary Block                          |  N/A  |    0   |
     +-----------------------------------------+-------+--------+
     |  Bundle Integrity Block                 |   11  |    2   |
     |  OP(bib-integrity, target=1)            |       |        |
     +-----------------------------------------+-------+--------+
     |  Payload Block                          |   0   |    1   |
     +-----------------------------------------+-------+--------+
```

Figure 3: Example 1 Resulting Bundle

A.1.3. Bundle Integrity Block

In this example, a BIB is used to carry an integrity signature over the payload block.

A.1.3.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BIB has a single target and includes a single security result: the calculated signature over the payload block.

```
            Key         : h'1a2b1a2b1a2b1a2b1a2b1a2b1a2b1a2b'
            SHA Variant : HMAC 512/512
            Scope Flags : 0
            Payload Data: h'52656164792047656e65726174652061
                            20333220627974652070617960c6f6164'
            Signature   : h'd8e7c3be29effa8779e7dcb0d3cadf53
                            39df50ebd27b9054f197c8ea9864b0a3
                            35a0636213e5d4a9c95504f261d91a2f
                            22757112c95e3587a76b4228361803e8'
```

Figure 4: Example 1: Configuration, Parameters, and Results

A.1.3.2. Abstract Security Block

The abstract security block structure of the BIB's block-type-specific-data field for this application is as follows.

```
[1],        / Security Target /
1,          / Security Context ID    - BIB-HMAC-SHA2       /
1,          / Security Context Flags - Parameters Present  /
[2,[2, 1]], / Security Source        - ipn:2.1             /
[           / Security Parameters    - 2 Parameters        /
   [1, 7],  / SHA Variant            - HMAC 512/512        /
   [3, 0]   / Scope Flags            - No Additional Scope /
],
[           / Security Results: 1 Result /
   [1, h'd8e7c3be29effa8779e7dcb0d3cadf5339df50ebd27b9054f197c8ea9864
         b0a335a0636213e5d4a9c95504f261d91a2f22757112c95e3587a76b4228
         361803e8']
]
```

Figure 5: Example 1: BIB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block-type-specific-data field (the abstract security block) is 0x810101018202820201828201078203008182015840d8e7c3be29effa8779e7dcb0d3cadf5339df50ebd27b9054f197c8ea9864b0a335a0636213e5d4a9c95504f261d91a2f22757112c95e3587a76b4228361803e8.

A.1.3.3. Representations

The BIB wrapping this abstract security block is as follows.

```
[
  11, / type code    /
  2,  / block number /
  0,  / flags        /
  0,  / CRC type     /
  h'810101018202820201828201078203008182015840d8e7c3be29effa8779e7dcb
    0d3cadf5339df50ebd27b9054f197c8ea9864b0a335a0636213e5d4a9c95504f2
    61d91a2f22757112c95e3587a76b4228361803e8',
]
```

Figure 6: Example 1: BIB (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block is 0x850b020000585581010101820282020182820107820300818201584 0d8e7c3be29effa8779e7dcb0d3cadf5339df50ebd27b9054f197c8ea9864b0a335a0636213e5d4a9c95504f261d91a2f22757112c95e3587a76b4228361803e8.

A.1.4. Final Bundle

The CBOR encoding of the full output bundle, with the BIB: 0x9F880700008202820102820282020182028202018200182 81a000f4240850b020000585581010101820282020182820107820300818201584 0d8e7c3be29effa8779e7dcb0d3cadf5339df50ebd27b9054f197c8ea9864b0a335a0636213e5d4a9c95504f261d91a2f22757112c95e3587a76b4228361803e8ff.

A.2. Example 2: Simple Confidentiality with Key Wrap

This example shows the addition of a BCB to a sample bundle to provide confidentiality for the payload block. AES key wrap is used to transmit the symmetric key used to generate the security results for this service.

A.2.1. Original Bundle

The following diagram shows the original bundle before the BCB has been added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Payload Block	0	1

Figure 7: Example 2 Original Bundle

A.2.1.1. Primary Block

The primary block used in this example is identical to the primary block presented in Example 1 Appendix A.1.1.1.

In summary, the CBOR encoding of the primary block is 0x88070000820282010282028202018202820201820018281a000f4240.

A.2.1.2. Payload Block

The payload block used in this example is identical to the payload block presented in Example 1 Appendix A.1.1.2.

In summary, the CBOR encoding of the payload block is 0x8501010000582 05265616479204765 6e65726174652061203332206279746520706179 6c6f6164.

A.2.1.3. Bundle CBOR Representation

A BPv7 bundle is represented as an indefinite-length array consisting of the blocks comprising the bundle, with a terminator character at the end.

The CBOR encoding of the original bundle is 0x9f88070000820282010282028202018202820201820018281a000f4240850101000058205265616479204765 6e65726174652061203332206279746520706179 6c6f6164ff.

A.2.2. Security Operation Overview

This example adds a BCB using the BCB-AES-GCM security context using AES key wrap to provide a confidentiality mechanism over the payload block and transmit the symmetric key.

The following diagram shows the resulting bundle after the BCB is added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Bundle Confidentiality Block OP(bcb-confidentiality, target=1)	12	2
Payload Block (Encrypted)	0	1

Figure 8: Example 2 Resulting Bundle

A.2.3. Bundle Confidentiality Block

In this example, a BCB is used to encrypt the payload block and uses AES key wrap to transmit the symmetric key.

A.2.3.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BCB has a single target, the payload block. Three security results are generated: cipher text which replaces the plain text block-type-specific data to encrypt the payload block, an authentication tag, and the AES wrapped key.

```
 Content Encryption
                Key: h'71776572747975696f70617364666768'
 Key Encryption Key: h'6162636465666768696a6b6c6d6e6f70'
                 IV: h'5477656c7665313231323132'
        AES Variant: A128GCM
    AES Wrapped Key: h'69c411276fecddc4780df42c8a2af892
                       96fabf34d7fae700'
        Scope Flags: 0
       Payload Data: h'52656164792047656e65726174652061
                       20333220627974652070617960c6f6164'
 Authentication Tag: h'689b98e649ae3b554e98aa2ae8f801eb'
 Payload Ciphertext: h'3a09c1e63fe2097528a78b7c12943354
                       a563e32648b700c2784e26a990d91f9d'
```

Figure 9: Example 2: Configuration, Parameters, and Results

A.2.3.2. Abstract Security Block

The abstract security block structure of the BCB's block-type-specific-data field for this application is as follows.

```
[1],        / Security Target /
2,          / Security Context ID    - BCB-AES-GCM        /
1,          / Security Context Flags - Parameters Present /
[2,[2, 1]], / Security Source        - ipn:2.1            /
[           / Security Parameters    - 4 Parameters       /
  [1, h'5477656c7665313231323132'], / Initialization Vector /
  [2, 1],                           / AES Variant - A128GCM /
  [3, h'69c411276fecddc4780df42c8a  / AES wrapped key /
        2af89296fabf34d7fae700'],
  [4, 0]                            / Scope Flags - No extra scope/
],
[           / Security Results: 1 Result /
  [1, h'689b98e649ae3b554e98aa2ae8f801eb'] / Payload Auth. Tag /
]
```

Figure 10: Example 2: BCB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block-type-specific-data field (the abstract security block) is 0x810102018202820201848201 4c5477656c76653132313231328202018203581869c411276fecddc4780df42c8a2af89296fabf34d7fae700820400818201506 89b98e649ae3b554e98aa2ae8f801eb.

A.2.3.3. Representations

The BCB wrapping this abstract security block is as follows.

```
[
  12, / type code /
  2,  / block number /
  1,  / flags - block must be replicated in every fragment /
  0,  / CRC type /
  h'810102018202820201848201 4c5477656c76653132313231328202018203581869c411276fecddc4780df42c8a2af89296fabf34d7fae700820400818201506 89b98e649ae3b554e98aa2ae8f801eb'
]
```

Figure 11: Example 2: BCB (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block is 0x850c020100584f810102018202820 2018482014c5477656c76653132313231328202018203581869c411276fecddc4780d f42c8a2af89296fabf34d7fae700820400818201506 89b98e649ae3b554e98aa2ae8f 801eb.

A.2.4. Final Bundle

The CBOR encoding of the full output bundle, with the BCB: 0x9f880700
008202820102820282020182028202018200182 81a000f4240850c020100584f81010
2018202820201848 2014c5477656c766531323132313282020182035 81869c411276f
ecddc4780df42c8a2af89296fabf34d7fae7008204008182015 0689b98e649ae3b554
e98aa2ae8f801eb850101000058203a09c1e63fe2097528a78b7c12943354a563e326
48b700c2784e26a990d91f9dff.

A.3. Example 3: Security Blocks from Multiple Sources

This example shows the addition of a BIB and BCB to a sample bundle. These two security blocks are added by two different nodes. The BCB is added by the source endpoint and the BIB is added by a forwarding node.

The resulting bundle contains a BCB to encrypt the Payload Block and a BIB to provide integrity to the Primary and Bundle Age Block.

A.3.1. Original Bundle

The following diagram shows the original bundle before the security blocks have been added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Extension Block: Bundle Age Block	7	2
Payload Block	0	1

Figure 12: Example 3 Original Bundle

A.3.1.1. Primary Block

The primary block used in this example is identical to the primary block presented in Example 1 Appendix A.1.1.1.

In summary, the CBOR encoding of the primary block is
0x88070000820282010282028202018202820201820018281a000f4240.

A.3.1.2. Bundle Age Block

A bundle age block is added to the bundle to help other nodes in the network determine the age of the bundle. The use of this block is as recommended because the bundle source does not have an accurate clock (as indicated by the DTN time of 0).

Because this block is specified at the time the bundle is being forwarded, the bundle age represents the time that has elapsed from the time the bundle was created to the time it is being prepared for forwarding. In this case, the value is given as 300 milliseconds.

The bundle age extension block is provided as follows.

```
[
  7,      / type code: Bundle Age block /
  2,      / block number /
  0,      / block processing flags /
  0,      / CRC Type /
  <<300>> / type-specific-data: age /
]
```

Figure 13: Bundle Age Block (CBOR Diagnostic Notation)

The CBOR encoding of the bundle age block is 0x85070200004319012c.

A.3.1.3. Payload Block

The payload block used in this example is identical to the payload block presented in Example 1 Appendix A.1.1.2.

In summary, the CBOR encoding of the payload block is 0x8501010000582
052656164792047656e65726174652061203332206279746520706179 6c6f6164.

A.3.1.4. Bundle CBOR Representation

A BPv7 bundle is represented as an indefinite-length array consisting of the blocks comprising the bundle, with a terminator character at the end.

The CBOR encoding of the original bundle is 0x9f88070000820282010282028202018202820201820018281a000f4240850702000043190 12c8501010000582052656164792047656e65726174652061203332206279746520706179 6c6f6164ff.

A.3.2. Security Operation Overview

This example provides:

a BIB with the BIB-HMAC-SHA2 security context to provide an integrity mechanism over the primary block and bundle age block.

a BCB with the BCB-AES-GCM security context to provide a confidentiality mechanism over the payload block.

The following diagram shows the resulting bundle after the security blocks are added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Bundle Integrity Block OP(bib-integrity, targets=0, 2)	11	3
Bundle Confidentiality Block OP(bcb-confidentiality, target=1)	12	4
Extension Block: Bundle Age Block	7	2
Payload Block (Encrypted)	0	1

Figure 14: Example 3 Resulting Bundle

A.3.3. Bundle Integrity Block

In this example, a BIB is used to carry an integrity signature over the bundle age block and an additional signature over the payload block. The BIB is added by a waypoint node, ipn:3.0.

A.3.3.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BIB has two security targets and includes two security results, holding the calculated signatures over the bundle age block and primary block.

```
                  Key: h'1a2b1a2b1a2b1a2b1a2b1a2b1a2b1a2b'
          SHA Variant: HMAC 256/256
          Scope Flags: 0
   Primary Block Data: h'88070000820282010282028202018202820201820 01
                         8281a000f4240'
   Bundle Age Block
                 Data: h'85070200004319012c'
   Primary Block
           Signature: h'2f74b42d88234f0a8a98a6c72775ec6511aff3cb5bf
                         c06aa648f5fc40f31ec0d'
   Bundle Age Block
           Signature: h'e61385353ce2b4cce5319bc33326cdc26f4061e76cb
                         21b434c89199a36b00de3'
```

Figure 15: Example 3: Configuration, Parameters, and Results for the BIB

A.3.3.2. Abstract Security Block

The abstract security block structure of the BIB's block-type-specific-data field for this application is as follows.

```
[0, 2],     / Security Target /
1,          / Security Context ID    - BIB-HMAC-SHA2       /
1,          / Security Context Flags - Parameters Present  /
[2,[3, 0]], / Security Source        - ipn:3.0             /
[           / Security Parameters    - 2 Parameters        /
   [1, 5],  / SHA Variant            - HMAC 256/256        /
   [3, 0]   / Scope Flags            - No Additional Scope /
],
[           / Security Results: 2 Results /
   [1, h'2f74b42d88234f0a8a98a6c72775ec6511aff3 / Primary Block    /
         cb5bfc06aa648f5fc40f31ec0d'],
   [1, h'e61385353ce2b4cce5319bc33326cdc26f4061 / Bundle Age Block /
         e76cb21b434c89199a36b00de3']
]
```

Figure 16: Example 3: BIB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block-type-specific-data field (the abstract security block) is 0x82000201018202820300828201058203008282 0158202f74b42d88234f0a8a98a6c72775ec6511aff3cb5bfc06aa648f5fc40f31ec0d 82015820e61385353ce2b4cce5319bc33326cdc26f4061e76cb21b434c89199a36b00 de3.

A.3.3.3. Representations

The BIB wrapping this abstract security block is as follows.

```
[
  11, / type code /
  3,  / block number /
  0,  / flags  /
  0,  / CRC type /
  h'82000201018202820300828201058203008282015820 2f74b42d88234f0a8a98
    a6c72775ec6511aff3cb5bfc06aa648f5fc40f31ec0d82015820e61385353ce2
    b4cce5319bc33326cdc26f4061e76cb21b434c89199a36b00de3',
]
```

Figure 17: Example 3: BIB (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block is 0x850b030000585a820002010182028
2030082820105820300828201582 02f74b42d88234f0a8a98a6c72775ec6511aff3cb
5bfc06aa648f5fc40f31ec0d82015820e61385353ce2b4cce5319bc33326cdc26f406
1e76cb21b434c89199a36b00de3.

A.3.4. Bundle Confidentiality Block

In this example, a BCB is used encrypt the payload block. The BCB is added by the bundle source node, ipn:2.1.

A.3.4.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BCB has a single target, the payload block. Two security results are generated: cipher text which replaces the plain text block-type-specific data to encrypt the payload block, and an authentication tag.

```
Content Encryption
                Key: h'71776572747975696f70617364666768'
                 IV: h'5477656c7665313231323132'
        AES Variant: A128GCM
        Scope Flags: 0
       Payload Data: h'52656164792047656e65726174652061
                       20333220627974652070617796c6f6164'
Authentication Tag: h'689b98e649ae3b554e98aa2ae8f801eb'
Payload Ciphertext: h'3a09c1e63fe2097528a78b7c12943354
                       a563e32648b700c2784e26a990d91f9d'
```

Figure 18: Example 3: Configuration, Parameters, and Results for the BCB

A.3.4.2. Abstract Security Block

The abstract security block structure of the BCB's block-type-specific-data field for this application is as follows.

```
[1],        / Security Target /
2,          / Security Context ID    - BCB-AES-GCM        /
1,          / Security Context Flags - Parameters Present /
[2,[2, 1]], / Security Source        - ipn:2.1            /
[           / Security Parameters    - 3 Parameters       /
  [1, b'Twelve121212'] / Initialization Vector /,
  [2, 1]               / AES Variant - AES 128 /,
  [4, 0]               / Scope Flags - No Additional Scope /
],
[           / Security Results: 1 Result /
  [1, h'689b98e649ae3b554e98aa2ae8f801eb'] / Payload Auth. Tag /
]
```

Figure 19: Example 3: BCB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block-type-specific-data field (the abstract security block) is 0x810102018202820201838201 4c5477656C76653 1323132313282020182040081820150689b98e649ae3b554e98aa2ae8f801eb.

A.3.4.3. Representations

The BCB wrapping this abstract security block is as follows.

```
[
  12, / type code /
  4,  / block number /
  1,  / flags - block must be replicated in every fragment /
  0,  / CRC type /
  h'810102018202820201838201 4c5477656C766531323132313282020182040081
    820150689b98e649ae3b554e98aa2ae8f801eb',
]
```

Figure 20: Example 3: BCB (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block is 0x850c0401005833810102018202820
2018382014c5477656C766531323132313282020182040081820150689b98e649ae3b
554e98aa2ae8f801eb.

A.3.5. Final Bundle

The CBOR encoding of the full output bundle, with the BIB and BCB added is: 9F88070000820282010282028202018202820201820018281a000f42408
50b030000585a82000201018202820300828201058203008282015820 2f74b42d8823
4f0a8a98a6c72775ec6511aff3cb5bfc06aa648f5fc40f31ec0d82015820e61385353
ce2b4cce5319bc33326cdc26f4061e76cb21b434c89199a36b00de3850c0401005833
810102018202820201838201 4c5477656C766531323132313282020182040081820 15
0689b98e649ae3b554e98aa2ae8f801eb85070200004319012c850101000058203a09
c1e63fe2097528a78b7c12943354a563e32648b700c2784e26a990d91f9dFF.

A.4. Example 4: Security Blocks with Full Scope

This example shows the addition of a BIB and BCB to a sample bundle. A BIB is added to provide integrity over the payload block and a BCB is added for confidentiality over the payload and BIB.

The integrity scope and additional authentication data will bind the primary block, target header, and the security header.

A.4.1. Original Bundle

The following diagram shows the original bundle before the security blocks have been added.

```
                          Block                    Block   Block
                        in Bundle                  Type    Number
        +=========================================+=======+========+
        |  Primary Block                          |  N/A  |    0   |
        +-----------------------------------------+-------+--------+
        |  Payload Block                          |   0   |    1   |
        +-----------------------------------------+-------+--------+
```

Figure 21: Example 4 Original Bundle

A.4.1.1. Primary Block

The primary block used in this example is identical to the primary block presented in Example 1 Appendix A.1.1.1.

In summary, the CBOR encoding of the primary block is 0x88070000820282010282028202018202820201820018281a000f4240.

A.4.1.2. Payload Block

The payload block used in this example is identical to the payload block presented in Example 1 Appendix A.1.1.2.

In summary, the CBOR encoding of the payload block is 0x8501010000582052656164792047656e65726174652061203332206279746520706179 6c6f6164.

A.4.1.3. Bundle CBOR Representation

A BPv7 bundle is represented as an indefinite-length array consisting of the blocks comprising the bundle, with a terminator character at the end.

The CBOR encoding of the original bundle is 0x9f88070000820282010282028202018202820201820018281a000f4240850101000058205265616479204765 6e65726174652061203332206279746520706179 6c6f6164ff.

A.4.2. Security Operation Overview

This example provides:

a BIB with the BIB-HMAC-SHA2 security context to provide an integrity mechanism over the payload block.

a BCB with the BCB-AES-GCM security context to provide a confidentiality mechanism over the payload block and BIB.

The following diagram shows the resulting bundle after the security blocks are added.

Block in Bundle	Block Type	Block Number
Primary Block	N/A	0
Bundle Integrity Block (Encrypted) OP(bib-integrity, target=1)	11	3
Bundle Confidentiality Block OP(bcb-confidentiality, targets=1, 3)	12	4
Payload Block (Encrypted)	0	1

Figure 22: Example 4 Resulting Bundle

A.4.3. Bundle Integrity Block

In this example, a BIB is used to carry an integrity signature over the payload block. The IPPT contains the payload block block-type-specific data, primary block data, the payload block header, and the BIB header. That is, all additional headers are included in the IPPT.

A.4.3.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BIB has a single target and includes a single security result: the calculated signature over the Payload block.

```
                     Key: h'1a2b1a2b1a2b1a2b1a2b1a2b1a2b1a2b'
             SHA Variant: HMAC 384/384
             Scope Flags: 7 (all additional headers)
       Primary Block Data: h'88070000820282010282028202018202
                            820201820018281a000f4240
             Payload Data: h'52656164792047656e65726174652061
                            20333220627974652070617 96c6f6164'
           Payload Header: h'85010100005820'
               BIB Header: h'850b0300005845'
        Payload Signature: h'6f56e0f58ec584df34603c75cc055939
                            00b1a938f23883f119772e1230441d86
                            9bce6ac9559f721260314424ab14b981
```

Figure 23: Example 4: Configuration, Parameters, and Results for the BIB

A.4.3.2. Abstract Security Block

The abstract security block structure of the BIB's block-type-specific-data field for this application is as follows.

```
[1],        / Security Target /
1,          / Security Context ID    - BIB-HMAC-SHA2 /
1,          / Security Context Flags - Parameters Present  /
[2,[2, 1]], / Security Source: ipn:2.1 /
[           / Security Parameters: 2 Parameters /
   [1, 6],  / SHA Variant - HMAC 384/384  /
   [3, 7]   / Scope Flags - All additional headers in the SHA Hash /
],
[           / Security Results: 1 Result /
   [1, h'6f56e0f58ec584df34603c75cc05593900b1a938f23883f119772e123044
         1d869bce6ac9559f721260314424ab14b981']
]
```

Figure 24: Example 4: BIB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block-type-specific-data field (the abstract security block) is 0x8101010182028202018282010682030781820158306f56e0f58ec584df34603c75cc05593900b1a938f23883f119772e1230441d869bce6ac9559f721260314424ab14b981.

A.4.3.3. Representations

The BIB wrapping this abstract security block is as follows.

```
[
  11, / type code /
  3,  / block number /
  0,  / flags  /
  0,  / CRC type /
  h'8101010182028202018282010682030781820158306f56e0f58ec584df34603c
    75cc05593900b1a938f23883f119772e1230441d869bce6ac9559f7212603144
    24ab14b981',
]
```

Figure 25: Example 4: BIB (CBOR Diagnostic Notation)

The CBOR encoding of the BIB block is 0x850b03000058458101010182028202018282010682030781820158306f56e0f58ec584df34603c75cc05593900b1a938f23883f119772e1230441d869bce6ac9559f721260314424ab14b981.

A.4.4. Bundle Confidentiality Block

In this example, a BCB is used encrypt the payload block and the BIB that provides integrity over the payload.

A.4.4.1. Configuration, Parameters, and Results

For this example, the following configuration and security parameters are used to generate the security results indicated.

This BCB has two targets: the payload block and BIB. Four security results are generated: cipher text which replaces the plain text block-type-specific data of the payload block, cipher text to encrypt the BIB, and authentication tags for both the payload block and BIB.

```
                     Key: h'71776572747975696f70617364666768
                            71776572747975696f70617364666768'
                      IV: h'5477656c7665313231323132'
              AES Variant: A256GCM
              Scope Flags: 7 (All additional headers)
             Payload Data: h'52656164792047656e65726174652061
                            20333220627974652070617796c6f6164'
                 BIB Data: h'52656164792047656E65726174652061
                            20333220627974652070617796C6F6164'
                     BIB
       Authentication Tag: h'92bc2665e9f04350c5974f023929dd62'
            Payload Block
       Authentication Tag: h'865bc14b3910d6c53e95fdc65aa601fd'
       Payload Ciphertext: h'90eab64575930498d6aa654107f15e96
                            319bb227706000abc8fcac3b9bb9c87e'
           BIB Ciphertext: h'438ed6208eb1c1ffb94d952175167df0
                            902a815f2276222e1d0208c628e2c926
                            2a0c438fc300190dbf5954ae4f84f748
                            64e58ed1e39043633142ad2559e0e3a9
                            c9cbce5c2d'
```

Figure 26: Example 4: Configuration, Parameters, and Results for the BCB

A.4.4.2. Abstract Security Block

The abstract security block structure of the BCB's block-type-specific-data field for this application is as follows.

```
[3, 1],     / Security Target /
2,          / Security Context ID    - BCB-AES-GCM /
1,          / Security Context Flags - Parameters Present  /
[2,[2, 1]], / Security Source        - ipn:2.1 /
[           / Security Parameters    - 3 Parameters /
  [1, h'5477656c7665313231323132'] / Initialization Vector /,
  [2, 3]               / AES Variant - AES 256 /,
  [4, 7]               / Scope Flags - All headers in SHA hash /
],
[           / Security Results: 2 Results /
  [1, h'865bc14b3910d6c53e95fdc65aa601fd'], / Payload Auth. Tag /
  [1, h'92bc2665e9f04350c5974f023929dd62']  / BIB Auth. Tag /
]
```

Figure 27: Example 4: BCB Abstract Security Block (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block-type-specific-data field (the abstract security block) is 0x82030102018202820201838201 4c5477656C766531323132313282020382040782820150d0b506cc2e5ede57b36e6c52791457008201 50865bc14b3910d6c53e95fdc65aa601fd.

A.4.4.3. Representations

The BCB wrapping this abstract security block is as follows.

```
[
  12, / type code /
  2,  / block number /
  1,  / flags - block must be replicated in every fragment /
  0,  / CRC type /
  h'82030102018202820201838201 4c5477656C7665313231323132820203820407
    82820150d0b506cc2e5ede57b36e6c527914570082015 0865bc14b3910d6c53e
    95fdc65aa601fd',
]
```

Figure 28: Example 4: BCB (CBOR Diagnostic Notation)

The CBOR encoding of the BCB block is 0x850c02010058478203010201820282 02018382014c5477656C7665313231323132820203820407828201 50d0b506cc2e5e de57b36e6c527914570082015 0865bc14b3910d6c53e95fdc65aa601fd.

A.4.5. Final Bundle

The CBOR encoding of the full output bundle, with the security blocks added and payload block and BIB encrypted is: 9F88070000820282010282 0282020182028202018200182 81a000f4240850b0300005845438ed6208eb1c1ffb94d 952175167df0902a815f2276222e1d0208c628e2c9262a0c438fc300190dbf5954ae4

f84f74864e58ed1e39043633142ad2559e0e3a9c9cbce5c2d 850c020100584782030
102018202820201838201 4c5477656C766531323132313282020382040782820150d0
b506cc2e5ede57b36e6c5279145700820150865bc14b3910d6c53e95fdc65aa601fd8
501010000582090eab64575930498d6aa654107f15e96319bb227706000abc8fcac3b
9bb9c87eFF.

Appendix B. Acknowledgements

The following participants contributed useful review and analysis of these security contexts: Amy Alford 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

Alex White
The Johns Hopkins University Applied
Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
US

Phone: +1 443 778 0845
Email: Alex.White@jhuapl.edu

Sarah Heiner
The Johns Hopkins University Applied
Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
US

Phone: +1 240 592 3704
Email: Sarah.Heiner@jhuapl.edu