BPSec Default Security Contexts
draft-ietf-dtn-bpsec-interop-sc-02

Abstract

This document defines default integrity and confidentiality security contexts that may be used with the Bundle Protocol Security Protocol (BPSec) implementations. These security contexts may 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.

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 [RFC7049].

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 [RFC4634] combined with the HMAC keyed hash discussed in [RFC2104].

BIB-HMAC-SHA2 supports three variants of HMAC-SHA, based on the supported length of the SHA-2 hash value. These variants correspond to HMAC256-SHA256, HMAC384-SHA384, and HMAC512-SHA512 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 shall 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 shall have the Security Context ID specified in Section 5.1.

3.2. Scope

The scope of BIB-HMAC-SHA2 refers to 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 contents 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) may 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 may 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 carrying the security result for this security context security block 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	HMAC256/SHA256 as defined in [RFC8152] Table 7: HMAC Algorithm Values
6	HMAC384/SHA384 as defined in [RFC8152] Table 7: HMAC Algorithm Values
7	HMAC512/SHA512 as defined in [RFC8152] Table 7: HMAC Algorithm Values

Table 1

When not provided, implementations SHOULD assume a value of 5 (indicating use of HMAC256/SHA256), unless an alternate default is established by security policy at the security source, verifier, or acceptor of this integrity service.

3.3.2. Encapsulated Key

This optional parameter contains the output of a Key Encapsulation Mechanism (KEM) run at the security source of this security context.

This value MUST be encoded as a CBOR byte string.

If provided, this information is used to retrieve the symmetric HMAC key used in the generation of security results for this security context. If not provided, security verifiers and acceptors MUST determine the proper key as a function of their local BPSec policy and configuration, as discussed in Section 3.5.

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 containing no more than 16 bits.

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, 0x1): Primary Block Flag.

- Bit 1 (0x02): Target Header Flag.

- Bit 2 (0x03): 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	Encoding Type	Default Value
1	SHA Variant	UINT	256
2	Encapsulated Key	Byte Array	NONE
3	Integrity Scope Flags	UINT	0

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

BIB-HMAC-SHA2 does not define or otherwise mandate any method for key exchange, encryption, or encapsulation. The derivation of an appropriate key for use in the integrity service is considered

separate from the application of the integrity service for this context.

HMAC keys used with this context MUST be symmetric and MUST have a key length equal to the output of the HMAC.

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 encapsulated key material carried in the BIB.

Session keys negotiated via a mechanism external to the BIB.

BIB-HMAC-SHA2 provides no explicit requirements on the configuration, storage, or exchange of HMAC keys.

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 may 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 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.

4. 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.

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 may 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.

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 may occur in any order.

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

Any local flags used to generated 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 encapsulated using some key encapsulation mechanism (to include encrypting with a key encryption key) and

the results of the encapsulation added as the Encapsulated 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 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 Encapsulated 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 if any needed parameters are missing then the verification MUST be treated as failed and processed in accordance with local security policy.

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].

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 may be specified via parameters to this security context.

This security context supports three variants of AES-GCM, based on the supported length of the symmetric key. These variants correspond to A128GCM, A192GCM, and A256GCM as defined in [RFC8152] Table 9: Algorithm Value for AES-GCM.

The BCB-AES-GCM security context shall have the Security Context ID 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 MUST include the plain text of the block-type-specific data field of the security target. This information 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) may 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 may 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 key length, 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 may 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 may be reused.

4.3.2. Key Length

This optional parameter identifies the key length being used for the AES-GCM encryption.

This value MUST be encoded as a CBOR unsigned integer.

Valid values for this parameter are as follows.

Key Length Parameter Values

Value	Description
1	A128GCM as defined in [RFC8152] Table 9: Algorithm Values for AES-GCM
2	A192GCM 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 security policy at the security source, verifier, or acceptor of this integrity service.

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

4.3.3. Encapsulated Key

This optional parameter contains the output of a Key Encapsulation

Mechanism (KEM) run at the security source of this security context.

This value MUST be encoded as a CBOR byte string.

If provided, this information is used to retrieve the symmetric AES key used in the generation of security results for this security context. If not provided, security verifiers and acceptors MUST determine the proper key as a function of their local BPSec policy and configuration, as discussed in Section 4.5.

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 containing no more than 16 bits.

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, 0x1): Primary Block Flag.

- Bit 1 (0x02): Target Header Flag.

- Bit 2 (0x03): 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	Encoding Type	Default Value
1	Initialization Vector	byte string	NONE
2	Key Length	UINT	3
3	Encapsulation Key	Byte Array	NONE
4	AAD Scope Flags	UINT	0

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

BCB-AES-GCM does not define or otherwise mandate any method for key exchange, encryption, or encapsulation. The derivation of an appropriate key is considered separate from the application of the authenticated confidentiality service provided by this context.

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.

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 encapsulated key material carried in the BCB.

Session keys negotiated via a mechanism external to the BCB.

BCB-AES-GCM provides no explicit requirements on the configuration, storage, or exchange of keys.

4.6. 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.

4.6.1. 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.

4.6.2. Additional Authenticated Data

The construction of additional authenticated data depends on the AAD Scope flags that may 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 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.

4. 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.

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

4.7. Processing

4.7.1. Encryption

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

The encryption key MUST have the appropriate length as required by local security policy. The key may 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 Initialization Vector (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 4.6.1.

Additional authenticated data, if present, MUST be generated as discussed in Section 4.6.2 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 may occur in any order.

The security context ID 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 encapsulated using some key encapsulation mechanism (to include encrypting with a key encryption key) and the results of the encapsulation added as the Encapsulated Key security parameter for the BCB.

The key length used by this security context MUST be added as the Key Length security parameter for the BCB if it differs from the default key length. Otherwise, the key length 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.

4.7.2. Decryption

During encryption, five inputs are prepared for input to the AES/GCM cipher: the decryption key, the Initialization Vector (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 Encapsulated 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 security policy at the decrypting node as discussed in Section 4.5.

The Initialization Vector (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 from local security policy in cases where IVs are predictable (such as always using an IV of 0 with constantly differing keys). Alternatively, a lack of an IV security parameter MAY be treated as an error by the decrypting node.

The security target cipher text for decryption MUST be generated as discussed in Section 4.6.1.

Additional authenticated data, if present, MUST be generated as discussed in Section 4.6.2 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 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

6. 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.

[I-D.ietf-dtn-bpbis] Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol Version 7", draft-ietf-dtn-bpbis-26 (work in progress), July 2020.

[I-D.ietf-dtn-bpsec] Birrane, E. and K. McKeever, "Bundle Protocol Security Specification", draft-ietf-dtn-bpsec-23 (work in progress), October 2020.

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

[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>.

[RFC4634] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, DOI 10.17487/RFC4634, July 2006, <https://www.rfc-editor.org/info/rfc4634>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[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>.

Appendix A. Acknowledgements

The following participants contributed useful review and analysis of these security contexts: Amy Alford and Sarah Heiner of the Johns Hopkins University Applied Physics Laboratory.

Author's Address

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