Delay-Tolerant Networking
Internet-Draft
Intended status: Standards Track
Expires: 18 April 2026

B. Sipos
JHU/APL

15 October 2025

Bundle Protocol Security (BPSec) COSE Context
draft-ietf-dtn-bpsec-cose-10

Abstract

This document defines a security context suitable for using CBOR Object Signing and Encryption (COSE) algorithms within Bundle Protocol Security (BPSec) integrity and confidentiality blocks. A profile for COSE, focused on asymmetric-keyed algorithms, and for PKIX certificates are also defined for BPSec interoperation.

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 18 April 2026.

Table of Contents

1. Introduction 3
1.1. Scope 3
1.2. PKIX Environments and CA Policy 4
1.3. Use of CDDL 4
1.4. Requirements Language 5
2. BPSec Security Context 5
2.1. Security Scope 5
2.2. Parameters 7
2.2.1. Additional Header Maps 8
2.2.2. AAD Scope 9
2.3. Results 10
2.3.1. Integrity Messages 11
2.3.2. Confidentiality Messages 12
2.4. Key Considerations 12
2.5. Canonicalization Algorithms 13
2.5.1. Generating AAD 13
2.5.2. Payload Data 15
2.6. Processing 15
2.6.1. Node Authentication 15
2.6.2. Policy Recommendations 16
3. COSE Profile 17
3.1. COSE Messages 17
3.2. Interoperability Algorithms 18
3.3. Needed Header Parameters 21
3.4. Symmetric Keys and Identifiers 23
3.5. Asymmetric Key Types and Identifiers 24
3.6. Policy Recommendations 25
4. PKIX Certificate Profile 25
4.1. Multiple-Certificate Uses 27
5. Security Considerations 27
5.1. Threat: BPSec Block Replay 27
5.2. Threat: Untrusted End-Entity Certificate 28
5.3. Threat: Certificate Validation Vulnerabilities 28
5.4. Threat: Security Source Impersonation 29
5.5. Threat: Unidentifiable Key 29
5.6. Threat: Non-Trusted Public Key 29
5.7. Threat: Passive Leak of Key Material 30
5.8. Threat: Algorithm Vulnerabilities 30
5.9. Inherited Security Considerations 30
5.10. AAD-Covered Block Modification 30
6. IANA Considerations 31
6.1. Bundle Protocol 31
7. References 32
7.1. Normative References 32
7.2. Informative References 34
Appendix A. Example Security Operations 35

A.1. Symmetric Key COSE_Mac0 38
A.2. EC Keypair COSE_Sign1 39
A.3. RSA Keypair COSE_Sign1 41
A.4. Symmetric CEK COSE_Encrypt0 44
A.5. Symmetric KEK COSE_Encrypt 46
A.6. EC Keypair COSE_Encrypt 48
A.7. RSA Keypair COSE_Encrypt 51
Appendix B. Example Public Key Certificates 55
B.1. Root CA Certificate 55
B.2. Signing Source End-Entity Certificate 57
B.3. Encryption Recipient End-Entity Certificate 58
Acknowledgments 60
Implementation Status 60
Author's Address 61

1. Introduction

The Bundle Protocol Security (BPSec) Specification [RFC9172] defines structure and encoding for Block Integrity Block (BIB) and Block Confidentiality Block (BCB) types but does not specify any security contexts to be used by either of the security block types. The CBOR Object Signing and Encryption (COSE) specifications [RFC9052] and [RFC9053] defines a structure, encoding, and algorithms to use for cryptographic signing and encryption.

This document describes how to use the algorithms and encodings of COSE within BPSec blocks to apply those algorithms to Bundle security in Section 2. A bare minimum of interoperability algorithms and algorithm parameters is specified by this document in Section 3. The focus of the recommended algorithms is to allow BPSec to be used in a Public Key Infrastructure (PKI) as described in Section 1.2 using a certificate profile defined in Section 4.

Examples of specific security operations are provided in Appendix A to aid in implementation support of the interoperability algorithms of Section 3.2. Examples of public key certificates are provided in Appendix B which are compatible with the profile in Section 4 and specific corresponding algorithms.

1.1. Scope

This document describes a profile of COSE which is tailored for use in BPSec and a method of including full COSE messages within BPSec security blocks. This document does not address:

* Policies or mechanisms for issuing Public Key Infrastructure Using X.509 (PKIX) certificates; provisioning, deploying, or accessing certificates and private keys; deploying or accessing certificate revocation lists (CRLs); or configuring security parameters on an individual entity or across a network.

* Uses of COSE beyond the profile defined in this document.

* How those COSE algorithms are intended to be used within a larger security context. Many header parameters used by COSE (e.g., key identifiers) depend on the network environment and security policy related to that environment.

1.2. PKIX Environments and CA Policy

This specification gives requirements about how to use PKIX certificates issued by a Certificate Authority (CA), but does not define any mechanisms for how those certificates come to be.

To support the PKIX uses defined in this document, the CA(s) issuing certificates for BP nodes are aware of the end use of the certificate, have a mechanism for verifying ownership of a Node ID, and are issuing certificates directly for that Node ID. BPSec security verifiers and acceptors authenticate the Node ID of security sources when verifying integrity (see Section 2.6.1) using a public key provided by a PKIX certificate (see Section 2.6.1) following the certificate profile of Section 4.

1.3. Use of CDDL

This document defines CBOR structure using the Concise Data Definition Language (CDDL) of [RFC8610]. The entire CDDL structure can be extracted from the XML version of this document using the XPath expression:

```
'//sourcecode[@type="cddl"]'
```

The following initial fragment defines the top-level symbols of this document's CDDL, including the ASB data structure with its parameter/result sockets.

```
start = bpsec-cose-asb / external_aad /
  primary-block / extension-block /
  MAC_structure / Sig_structure / Enc_structure / COSE_KeySet
```

From the document [RFC9052] the definitions are taken for MAC_structure, Sig_structure, Enc_structure, and COSE_KeySet. From the document [RFC9171] the definitions are taken for eid, primary-block, and extension-block.

1.4. 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.

2. BPSec Security Context

This document specifies a single security context for use in both BPSec integrity and confidentiality blocks. This is done to save code points allocated to this specification and to simplify the encoding of COSE-in-BPSec; the BPSec block type uniquely defines the acceptable parameters and COSE messages which can be present.

The COSE security context SHALL have the Security Context ID specified in Section 6.1.

Both types of security block can use the same parameters, defined in Section 2.2, to carry public key-related information and each type of security block allows specific COSE message results, defined in Section 2.3.

```
; Specialize the ASB for this context
bpsec-cose-asb = bpsec-context-use<
  3, ; Context ID COSE
  $bpsec-cose-param,
  $bpsec-cose-result
>
$ext-data-asb /= bpsec-cose-asb
```

Figure 1: COSE context declaration CDDL

2.1. Security Scope

The scope here refers to the set of information used by the security context to cryptographically bind with the plaintext data being integrity-protected or confidentiality-protected. This information is generically referred to as additional authenticated data (AAD), which is also the term used by COSE to describe the same data.

The sources for AAD within the COSE context are described below, controlled by the AAD Scope parameter (Section 2.2.2), and implemented as defined in Section 2.5.1. The purpose of this parameter is similar to the AAD Scope parameter and Integrity Scope parameter of [RFC9173] but expanded to allow including _any_ block in the bundle as AAD.

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 target is no longer in its original bundle. Including the primary block as part of AAD ensures that security target appears in the same bundle that the security source intended.

Canonical Block-Type-Specific Data:
: Including the block-type-specific data (BTSD) of a non-target block as part of AAD ensures that that other block's BTSD does not change after the security block is added. This can guarantee that not only has the security target BTSD not changed but the additional blocks' BTSD have not changed.

Canonical Block Metadata:
: Including block metadata, which identifies and types a block, as part of AAD ensures that the block presence does not change after the security block is added. This metadata explicitly excludes the CRC type and value fields because the CRC is derived from the BTSD. The metadata of the security block and the target block are also allowed, which binds the security result to that specific target.

Containing Abstract Security Block:
: The above sources of AAD allow covering the primary block, target block, and any other block besides the security block which contains the security operation for this COSE context The metadata of the containing security block can be included as described above, using AAD scope key -2 with the flag for metadata, but the BTSD of the security block (as defined in Section 3.6 of [RFC9172]) is also partially covered by AAD.

  The Security Targets field can be included indirectly by using AAD scope key -1 with the flag for metadata, which includes the target block number. The Security Context ID is not included directly, but modification of this field will cause processing (verification or acceptance) of the associated security operations to fail. The Security Source field is included as AAD unconditionally, so is protected from modification. The Security Context Flags and Security Context Parameters are not included directly, but the

modification of parameters will cause processing of security operations to fail. The Security Results are also not included directly, but these are the COSE messages themselves which are designed to be handled as plaintext.

Because of the above, it is possible for a security source to create a COSE context integrity operation which covers every block of a bundle at the time the BIB is added (excluding CRC Type and value fields). By using a minimal AAD scope it is also possible for an integrity operation to cover only the BTSD of a single target block independently of the block metadata or bundle primary block associated with the target at the time the BIB is added.

2.2. Parameters

Each COSE context parameter value SHALL consist of the COSE structure indicated by Table 1 in its decoded (CBOR item) form. Each security block SHALL contain no more than one of each parameter type per target block.

Parameter ID	Parameter Structure	Reference
3	additional-protected	Section 2.2.1 of this document
4	additional-unprotected	Section 2.2.1 of this document
5	AAD-scope	Section 2.2.2 of this document

Table 1: COSE Context Parameters

When a parameter is not present and a default value is defined below, a security verifier or acceptor SHALL use that default value to process the target:

* The default additional-protected is ’’ (an empty byte string).

* The default additional-unprotected is ’’ (an empty byte string).

* The default AAD-scope is {0:0b1,-1:0b1,-2:0b1} (a map which indicates the AAD contains the metadata of the primary, target, and security blocks).

2.2.1. Additional Header Maps

The two parameters Additional Protected and Additional Unprotected allow de-duplicating header items which are common to all COSE results. Both additional header values contain a CBOR map which is to be merged with each of the result's unprotected headers. Although the additional header items are all treated as unprotected from the perspective of the COSE message, the additional protected map is included within the external_aad (see Section 2.5.1). The expected use of additional header map is to contain a certificate (chain) or identifier (see Section 3.5) which applies to all results in the same security block.

Following the same pattern as COSE, when both additional header maps are present in a single security block they SHALL not contain any duplicated labels. Security verifiers and acceptors SHALL treat a pair of additional header maps containing duplicated labels as invalid.

No more than one of each Additional Protected and Additional Unprotected parameter SHALL be present in a single security block. Security verifiers and acceptors SHALL treat a security block with multiple instances of either additional header type as invalid. There is no well-defined behavior for a security acceptor to handle multiple Additional Protected parameters.

Security sources SHOULD NOT include an additional header parameter which represents an empty map. Security verifiers and acceptors SHALL handle empty header map parameters, specifically the Additional Protected parameter because it is part of the external_aad.

Security verifiers and acceptors SHALL treat the aggregate of both additional header maps as being present in the unprotected header map of the highest-layers of the COSE message of each result. For single-layer messages (i.e., COSE_Encrypt0, COSE_MAC0, and COSE_Sign1) the additional headers apply to the message itself (layer 0) and for other messages the additional headers apply to the final recipients. If the same header label is present in a additional header map and a COSE layer's headers the item in the result header SHALL take precedence (i.e., the additional header items are added only if they are not already present in a layer's header).

Additional header maps SHALL NOT contain any private key material. The security parameters are all stored in the bundle as plaintext and are visible to any bundle handlers.

```
$bpsec-cose-param /= [3, additional-protected]
additional-protected = empty_or_serialized_map

$bpsec-cose-param /= [4, additional-unprotected]
additional-unprotected = empty_or_serialized_map
```

Figure 2: Additional Headers CDDL

2.2.2. AAD Scope

The AAD Scope parameter controls what data is included in the AAD for both integrity and confidentiality operations. The AAD Scope parameter SHALL be encoded as a CBOR map containing keys referencing bundle blocks (as int items) and values representing a collection of bit flags (as uint items) defined in Table 2.

All non-negative AAD Scope keys SHALL correspond with block numbers in the bundle containing the AAD Scope parameter. Security verifiers and acceptors SHALL treat any AAD Scope with block numbers not actually present in the containing bundle as invalid. The AAD Scope key -1 SHALL be interpreted as corresponding to the target block of the security operation when the AAD is generated from the AAD Scope parameter. The AAD Scope key -2 SHALL be interpreted as corresponding to the security block which contains the AAD Scope parameter.

Bit Position (from LSbit)	Name	Description
0	AAD-metadata	If bit is set, indicates that the block metadata is included in the AAD.
1	AAD-btsd	If bit is set, indicates that the BTSD is included in the AAD.

Table 2: AAD Scope Flags

Any AAD Scope value bits SHALL NOT all be set to zero, which would represent the lack of presence in the AAD and serves no purpose. When the map key identifies the primary block (block number zero) the bits SHALL only have AAD-metadata set, as the primary block has no BTSD. When the map key identifies the containing security block the bits SHALL only have AAD-metadata set, as the security block BTSD

does not yet exist. When the map key identifies the target block the bits SHALL only have AAD-metadata set, as the target block BTSD is already part of the security operation (integrity or confidentiality). All unassigned bits SHALL be set to zero (which will be elided by CBOR encoding) by security sources. All unassigned bits SHALL be ignored by security verifiers and acceptors.

A CDDL representation of this definition is included in Figure 3 for reference.

```
$bpsec-cose-param /= [5, AAD-scope]
AAD-scope = {
    *blk-id => (uint .bits AAD-scope-flags)
}
blk-id = uint / blk-target / blk-sec
blk-target = -1
blk-sec = -2
AAD-scope-flags = &(
    AAD-metadata: 0,
    AAD-btsd: 1,
)
```

Figure 3: AAD Scope CDDL

The default value for this parameter (in Section 2.2) includes the primary, target, and security block metadata.

2.3. Results

Although each COSE context result is a COSE message, the types of message allowed depend upon the security block type in which the result is present: only MAC or signature messages are allowed in a BIB and only encryption messages are allowed in a BCB.

The code points for Result ID values are identical to the existing COSE message-marking tags in Section 2 of [RFC9052]. This avoids the need for value-mapping between code points of the two registries.

When embedding COSE messages, the message CBOR structure SHALL be encoded as a byte string used as the result value. This allows a security acceptor to skip over unwanted results without needing to decode the result structure. When embedding COSE messages, the CBOR-tagged form SHALL NOT be used. The Result ID values already provide the same information as the COSE tags (using the same code points).

These generic requirements are formalized in the CDDL fragment of Figure 4.

```
$bpsec-cose-result /= [16, bstr .cbor COSE_Encrypt0]
$bpsec-cose-result /= [17, bstr .cbor COSE_Mac0]
$bpsec-cose-result /= [18, bstr .cbor COSE_Sign1]
$bpsec-cose-result /= [96, bstr .cbor COSE_Encrypt]
$bpsec-cose-result /= [97, bstr .cbor COSE_Mac]
$bpsec-cose-result /= [98, bstr .cbor COSE_Sign]
```

Figure 4: COSE context results CDDL

2.3.1. Integrity Messages

When used within a Block Integrity Block, the COSE context SHALL allow only the Result IDs from Table 3. Each integrity result value SHALL consist of the COSE message indicated by Table 3 in its non-tagged encoded form.

Result ID	Result Structure	Reference
97	encoded COSE_Mac	[RFC9052]
17	encoded COSE_Mac0	[RFC9052]
98	encoded COSE_Sign	[RFC9052]
18	encoded COSE_Sign1	[RFC9052]

Table 3: COSE Integrity Results

Each integrity result SHALL use the "detached" payload form with null payload value. The integrity result for COSE_Mac and COSE_Mac0 messages are computed by the procedure in Section 6.3 of [RFC9052]. The integrity result for COSE_Sign and COSE_Sign1 messages are computed by the procedure in Section 4.4 of [RFC9052].

The COSE "protected attributes from the application" used for a signature or MAC result SHALL be the encoded data defined in Section 2.5.1. The COSE payload used for a signature or MAC result SHALL be either the BTSD of the target, if the target is not the primary block, or an empty byte string if the target is the primary block.

2.3.2. Confidentiality Messages

When used within a Block Confidentiality Block, COSE context SHALL allow only the Result IDs from Table 4. Each confidentiality result value SHALL consist of the COSE message indicated by Table 4 in its non-tagged encoded form.

Result ID	Result Structure	Reference
96	encoded COSE_Encrypt	[RFC9052]
16	encoded COSE_Encrypt0	[RFC9052]

Table 4: COSE Confidentiality Results

Only algorithms which support Authenticated Encryption with Authenticated Data (AEAD) SHALL be usable in the first (content) layer of a confidentiality result. Because COSE encryption with AEAD appends the authentication tag with the ciphertext, the size of the BTSD will grow after an encryption operation. Security verifiers and acceptors SHALL NOT assume that the size of the plaintext is the same as the size of the ciphertext.

Each confidentiality result SHALL use the "detached" payload form with null payload value. The confidentiality result for COSE_Encrypt and COSE_Encrypt0 messages are computed by the procedure in Section 5.3 of [RFC9052].

The COSE "protected attributes from the application" used for an encryption result SHALL be the encoded data defined in Section 2.5.1. The COSE payload used for an encryption result SHALL be the BTSD of the target. Because confidentiality of the primary block is disallowed by BPSec, there is no logic here for handling a BCB with a target on the primary block.

2.4. Key Considerations

This specification does not impose any additional key requirements beyond those already specified for each COSE algorithm required in Section 3.

2.5. Canonicalization Algorithms

Generating or processing COSE messages for the COSE context follows the profile defined in Section 3 with the "protected attributes from the application" (i.e., the external_aad item) generated as defined in Section 2.5.1.

2.5.1. Generating AAD

The AAD contents and encoding defined in this section are used for both integrity and confidentiality messages. The encoding of this AAD is different from AAD of Section 4.7.2 of [RFC9173] and the front items of IPPT of Section 3.7 of [RFC9173] due to support for AAD covering the ASB security source field and covering an arbitrary number of blocks in the same bundle.

When used as the external_aad for COSE operations, the AAD SHALL be encoded in accordance with the core deterministic encoding requirements of Section 4.2.1 of [RFC8949]. The AAD byte string SHALL consist of an encoded CBOR sequence, generated by concatenating the following byte string parts:

1. The first part SHALL be the encoded Security Source EID associated with the ASB containing this security operation.

2. The second part SHALL be the encoded AAD Scope value itself, which is a CBOR map. Because of deterministic encoding, the negative keys will occur after positive keys.

3. For each entry of the AAD Scope map, in ascending block number order followed by the negative sentinel values in descending order, the next items SHALL be one or both of the following:

    a. If the map value has the AAD-metadata flag set, the next part is block metadata taken from either:

        * If the map key is block number zero, the next part SHALL be the encoded form of the primary block of the containing bundle. This represents the full primary block, including its definite-length array head.

        * Otherwise, next part SHALL be the encoded form of the first three fields of the block (_i.e._, the block type code, block number, and control flags) identified by the block number in the map key. This part is just the three encoded integer fields concatenated with no framing (array or otherwise).

   b. If the map value has the AAD-btsd flag set and the map key is _not_ block number zero, the next part is the encoded BTSD of the block identified by the block number in the map key. This part includes a byte string head.

4. The last part SHALL be the encoded form of the Additional Protected parameter. This part includes a byte string head. This has a default value of an empty string, defined in Section 2.2.

Be aware that, because of deterministic encoding requirements here, there is no guarantee that AAD parts containing the same CBOR data as the ASB or containing bundle (_e.g._, the Security Source field), result in the same encoded byte string. When generated by the same entity they are expected to be the same, but an entity verifying or accepting a security operation SHALL treat bundle and block contents as untrusted input and re-encode the AAD parts.

A CDDL representation of this data is shown below in Figure 5.

```
; Specialized here to contain a specific sequence
external_aad /= bstr .cborseq AAD-list

AAD-list = [
    security-source: eid,
    AAD-scope,
    *AAD-block,
    ; copy of additional-protected (or default empty bstr)
    additional-protected
]
; each AAD item is a group, not a sub-array
AAD-block = (
    ? primary-block,   ; present for block number zero
    ? block-metadata,  ; present if AAD-metadata flag set
    ? bstr,            ; present if AAD-btsd flag set
)
; Selected fields of a canonical block
block-metadata = (
    block-type-code: uint,
    block-number: uint,
    block-control-flags,
)
```

Figure 5: COSE context AAD CDDL

2.5.2. Payload Data

When correlating between BPSec target BTSD and COSE plaintext or payload, any byte string SHALL be handled in its decoded (CBOR item) form. This means any CBOR header or tag in a source encoding are ignored for the purposes of security processing. This also means that if the source byte string was encoded in a non-conforming way, for example in indefinite-length form or with a non-minimum-size length, the security processing always treats it in a deterministically encoded CBOR form.

2.6. Processing

This section describes block-level requirements for handling COSE security data.

All security results generated for BIB or BCB blocks SHALL conform to the COSE profile of Section 3 with header augmentation as defined in Section 2.2.1.

2.6.1. Node Authentication

This section explains how the certificate profile of Section 4 is used by a security acceptor to both validate an end-entity certificate and to use that certificate to authenticate the security source for an integrity result. For a confidentiality result, some of the requirements in this section are implicit in an implementation using a private key associated with a certificate used by a result recipient. It is an implementation matter to ensure that a BP agent is configured to generate or receive results associated with valid certificates.

A security source MAY prohibit generating a result (either integrity or confidentiality) for an end-entity certificate which is not considered valid according to Section 2.6.1.2. Generating a result which is likely to be discarded is wasteful of bundle size and transport resources.

2.6.1.1. Certificate Identification

Because of the standard policy of using separate certificates for transport, signing, and encryption (see Section 4.1) a single Node ID is likely to be associated with multiple certificates, and any or all of those certificates MAY be present within an "x5bag" in an Additional Protected parameter (see Section 2.2.1). When present, a security verifier or acceptor SHALL use an "x5chain" or "x5t" to identify an end-entity certificate to use for result processing. Security verifiers and acceptors SHALL NOT assume that a validated

certificate containing a NODE-ID matching a security source is enough to associate a certificate with a COSE message or recipient (see Section 3.5).

2.6.1.2. Certificate Validation

For each end-entity certificate contained in or identified by by a COSE result, a security verifier or acceptor SHALL perform the certification path validation of Section 6 of [RFC5280] up to one of the acceptor's trusted CA certificates. When evaluating a certificate Validity time interval, a security verifier or acceptor SHALL use the Bundle Creation Time of the primary block as the reference instead of the current time. If enabled by local policy, the entity SHALL perform an OCSP check of each certificate providing OCSP authority information in accordance with [RFC6960]. If certificate validation fails or if security policy disallows a certificate for any reason, the acceptor SHALL treat the associated security result as failed. Leaving out part of the certification chain can cause the entity to fail to validate a certificate if the left-out certificates are unknown to the entity (see Section 5.2).

For each end-entity certificate contained in or identified by a COSE context result, a security verifier or acceptor SHALL apply security policy to the Key Usage extension (if present) and Extended Key Usage extension (if present) in accordance with Section 4.2.1.12 of [RFC5280] and the profile in Section 4.

2.6.1.3. Node ID Authentication

If required by security policy, for each end-entity certificate referenced by a COSE context integrity result a security verifier or acceptor SHALL validate the certificate NODE-ID in accordance with Section 6 of [RFC6125] using the NODE-ID reference identifier from the Security Source of the containing security block. If the NODE-ID validation result is Failure or if the result is Absent and security policy requires an authenticated Node ID, a security verifier or acceptor SHALL treat the result as failed.

2.6.2. Policy Recommendations

A RECOMMENDED security policy is to enable the use of OCSP checking when internet connectivity is present. A RECOMMENDED security policy is that if an Extended Key Usage is present that it needs to contain id-kp-bundleSecurity of [IANA-SMI] to be usable as an end-entity certificate for COSE security results. A RECOMMENDED security policy is to require a validated Node ID (of Section 2.6.1.3) and to ignore any other identifiers in the end-entity certificate.

This policy relies on and informs the certificate requirements in Section 3.6 and Section 4. This policy assumes that a DTN-aware CA (see Section 1.2) will only issue a certificate for a Node ID when it has verified that the private key holder actually controls the DTN node; this is needed to avoid the threat identified in Section 5.4. This policy requires that a certificate contain a NODE-ID and allows the certificate to also contain network-level identifiers. A tailored policy on a more controlled network could relax the requirement on Node ID validation and/or Extended Key Usage presence.

3. COSE Profile

This section contains requirements which apply to the use of COSE within the BPSec security context defined in this document. Other variations of COSE within BPSec can be used but are not expected to be interoperable or usable by all security verifiers and acceptors.

3.1. COSE Messages

When generating a BPSec result, security sources SHALL use only COSE labels with a uint value. When processing a BPSec result, security verifiers and acceptors MAY handle COSE labels with with a tstr value.

When used in a BPSec result, each COSE message SHALL contain an explicit algorithm identifier in the first (content) layers. When available and not implied by the bundle source, a COSE message SHALL contain a key identifier in the last (recipient) layer. See Section 3.5 for specifics about asymmetric key identifiers. When a key identifier is not available, BPSec acceptors SHALL use the Security Source (if available) and the Bundle Source to imply which keys can be used for security operations. Using implied keys has an interoperability risk, see Section 5.5 for details. A BPSec security operation always occurs within the context of the immutable primary block with its parameters (specifically the Source Node ID) and the security block with its optional Security Source.

The algorithms required by this profile focuses on networks using shared symmetric-keys, with recommended algorithms for Elliptic Curve (EC) keypairs and RSA keypairs. The focus of this profile is to enable interoperation between security sources and acceptors on an open network, where more explicit COSE parameters make it easier for BPSec acceptors to avoid assumptions and avoid out-of-band parameters. The requirements of this profile still allow the use of potentially not-easily-interoperable algorithms and message/recipient configurations for use by private networks, where message size is more important than explicit COSE parameters.

3.2. Interoperability Algorithms

The set of COSE algorithms needed for interoperability is listed in this section. The full set of COSE algorithms available is managed at [IANA-COSE].

Implementations conforming to this specification SHALL support the symmetric keyed and key-encryption algorithms marked as "required" in Table 5. Implementations capable of doing so SHOULD support the asymmetric keyed and key-encryption algorithms marked as "recommended" in Table 5.

> All of these algorithms in this table are compatible with FIPS-140-3 [FIPS-140].
>
> The required algorithms are identical to the capability of the [RFC9173] contexts.
>
> The EC-based algorithms are CNSA 1.0 conformant [CNSA1] only when used with a key having curve P-384. The RSA-based algorithms are CNSA 1.0 conformant [CNSA1] only when used with a key modulus of 3072 bits or larger. Only some symmetric-keyed algorithms are CNSA 2.0 conformant [CNSA2], as no quantum resistant public key algorithms are included in this profile.

BPSec Block	COSE Layer	Name	Code	Implementation Requirements	Conformance
Integrity	0 or 1	HMAC 256/256	5	Required	
Integrity	0 or 1	HMAC 384/384	6	Required	CNSA 1.0
Integrity	0 or 1	HMAC 512/512	7	Required	CNSA 1.0
Integrity	0 or 1	ESP256	-9	Recommended	
Integrity	0 or 1	Ed25519	-19	Recommended	
Integrity	0 or 1	ESP384	-51	Recommended	CNSA 1.0
Integrity	0 or 1	ESP512	-52	Recommended	CNSA 1.0
Integrity	0 or 1	PS256	-37	Recommended	

Integrity	0 or 1	PS384	-38	Recommended	CNSA 1.0
Integrity	0 or 1	PS512	-39	Recommended	CNSA 1.0
Integrity	0 or 1	Ed448	-53	Recommended	
Confidentiality	0	A128GCM	1	Required	
Confidentiality	0	A192GCM	2	Required	
Confidentiality	0	A256GCM	3	Required	CNSA 1.0 and 2.0
Confidentiality	1	A128KW	-3	Required	
Confidentiality	1	A192KW	-4	Required	
Confidentiality	1	A256KW	-5	Required	CNSA 1.0 and 2.0
Confidentiality	1	direct	-6	Recommended	
Confidentiality	1	ECDH-ES + HKDF-256	-25	Recommended	
Confidentiality	1	ECDH-ES + HKDF-512	-26	Recommended	CNSA 1.0
Confidentiality	1	ECDH-SS + HKDF-256	-27	Recommended	
Confidentiality	1	ECDH-SS + HKDF-512	-28	Recommended	CNSA 1.0
Confidentiality	1	ECDH-ES + A256KW	-31	Recommended	CNSA 1.0
Confidentiality	1	ECDH-SS + A256KW	-34	Recommended	CNSA 1.0
Confidentiality	1	RSAES-OAEP w/	-41	Recommended	

		SHA-256			
Confidentiality	1	RSAES-OAEP w/ SHA-512	-42	Recommended	CNSA 1.0
Any	x5t parameters	SHA-256/64	-15	Required	
Any	x5t parameters	SHA-256	-16	Required	
Any	x5t parameters	SHA-512/256	-17	Recommended	
Any	x5t parameters	SHA-384	-43	Recommended	CNSA 1.0 and 2.0
Any	x5t parameters	SHA-512	-44	Recommended	CNSA 2.0

Table 5: Interoperability Algorithms

The following are recommended key and recipient uses within COSE/ BPSec:

Symmetric Key Integrity:
When generating a BIB result from a symmetric key, implementations SHALL use a COSE_Mac or COSE_Mac0 using the private key directly. When a COSE_Mac or COSE_Mac0 is used with a direct key, the top-layer headers SHALL include a key identifier (see Section 3.4).

EC Keypair Integrity:
When generating a BIB result from an EC keypair, implementations SHALL use a COSE_Sign or COSE_Sign1 using the private key directly. When a COSE_Sign or COSE_Sign1 is used with an EC keypair, the top-layer headers SHALL include a public key identifier (see Section 3.5).

RSA Keypair Integrity:
When generating a BIB result from an RSA keypair, implementations SHALL use a COSE_Sign or COSE_Sign1 using the private key directly. When a COSE_Sign or COSE_Sign1 is used with an RSA keypair, the top-layer headers SHALL include a public key identifier (see Section 3.5). When a COSE signature is generated with an RSA keypair, the signature uses a PSS salt in accordance with Section 2 of [RFC8230].

Symmetric Key Confidentiality:
When generating a BCB result from a symmetric key-encryption key (KEK), implementations SHOULD use a COSE_Encrypt message with a recipient containing an indirect (wrapped or derived) content encryption key (CEK). When a COSE_Encrypt is used with an overall KEK, the recipient layer SHALL include a key identifier for the KEK.

When generating a BCB result from a symmetric CEK, implementations SHOULD use COSE_Encrypt or COSE_Encrypt0 with direct CEK. Session CEKs SHALL be managed to avoid overuse and the vulnerabilities associated with large amount of ciphertext from the same key.

EC Keypair Confidentiality:
When generating a BCB result from an EC public key, implementations SHALL use a COSE_Encrypt message with a recipient containing an indirect (wrapped or derived) CEK. When a COSE_Encrypt is used with an EC public key, the recipient layer SHALL include a public key identifier (see Section 3.5). When a COSE_Encrypt is used with an EC public key, the security source SHALL either generate an ephemeral EC keypair or choose a unique HKDF "salt" for each security operation. When a COSE_Encrypt is used with an EC public key and a single recipient, the direct HKDF algorithms (code -25 and -27) are RECOMMENDED over the key wrapped algorithms (code -31 and -34) to reduce message size. When processing a COSE_Encrypt with an EC public key, a security verifier or acceptor SHALL process all KDF and HMAC context data from the recipient headers in accordance with Section 5.2 of [RFC9053] even though the source is not required to provide any of those parameters.

RSA Keypair Confidentiality:
When generating a BCB result from an RSA public key, implementations SHALL use a COSE_Encrypt message with a recipient containing a key-wrapped CEK. When a COSE_Encrypt is used with an RSA public key, the recipient layer SHALL include a public key identifier (see Section 3.5).

Implementations conforming to this specification SHALL support the hash algorithms SHA-256 (code -16) and SHA-256/64 (code -15). Implementations SHOULD support the hash algorithms SHA-384 (code -43), SHA-512 (code -44), and SHA-512/256 (code -17).

3.3. Needed Header Parameters

The set of COSE header parameters needed for interoperability is listed in this section. The full set of COSE header parameters available is managed at [IANA-COSE].

Implementations conforming to this specification SHALL support the header parameters in Table 6. This support means required-to-implement not required-to-use for any particular COSE message.

Specific COSE algorithms have their own requirements about which header parameters are mandatory or optional to use in the associated COSE message layer. The phrasing in Table 6 uses the term "optional" where the need for a parameter is determined by the specific end use, and "conditional" for cases where one parameter of several options is needed by this profile. For example, a choice of specific symmetric key identifier (Section 3.4) or asymmetric key identifier (Section 3.5) among several options.

Name	Label	Need
alg	1	Required by COSE [RFC9052]
crit	2	Conditional for COSE [RFC9052]
content type	3	Optional for COSE [RFC9052]
kid	4	Conditional for this COSE profile
IV	5	Conditional for symmetric encryption algorithms
Partial IV	6	Conditional for symmetric encryption algorithms
kid context	10	Optional for this COSE profile
x5bag	32	Conditional for public key algorithms
x5chain	33	Conditional for public key algorithms
x5t	34	Conditional for public key algorithms
ephemeral key	-1	Required for ECDH-ES algorithms
static key	-2	Conditional for ECDH-SS algorithms
static key id	-3	Conditional for ECDH-SS algorithms

salt	-20	Conditional for HKDF and ECDH algorithms
PartyU identity	-21	Optional for HKDF and ECDH algorithms
PartyU nonce	-22	Optional for HKDF and ECDH algorithms
PartyU other	-23	Optional for HKDF and ECDH algorithms
PartyV identity	-24	Optional for HKDF and ECDH algorithms
PartyV nonce	-25	Optional for HKDF and ECDH algorithms
PartyV other	-26	Optional for HKDF and ECDH algorithms
x5t-sender	-27	Optional for ECDH-SS algorithms
x5chain-sender	-29	Optional for ECDH-SS algorithms

Table 6: Interoperability Header Parameters

3.4. Symmetric Keys and Identifiers

This section applies when a BIB or BCB uses a shared symmetric key for MAC, encryption, or key-wrap. When using symmetric keyed algorithms, the security source SHALL include a symmetric key identifier as a signature or recipient header. The symmetric key identifier SHALL be either a "kid" of [RFC9052] (possibly with "kid context" of [RFC8613]), or an equivalent identifier. This requirement makes the selection of keys by verifiers and acceptors unambiguous.

When present, a "kid" parameter is used to uniquely identify a single shared key known to the security source and all expected security verifiers and acceptors. Specific strategies or mechanisms to generate or ensure uniqueness of "kid" values within some domain of use is outside the scope of this profile. Specific users of this profile can define such mechanisms specific to their abilities and needs.

When present, a "kid context" parameter SHALL be used as a correlator with a larger scope than an individual "kid" value. The use of a "kid context" allows security verifiers and acceptors to correlate using that larger scope even if they cannot match the sibling "kid" value. For example, a "kid context" can be used to identify a long-lived security association between two entities while an individual "kid" identifies a single shared key agreed within that larger association.

3.5. Asymmetric Key Types and Identifiers

This section applies when a BIB uses a public key for verification or key-wrap, or when a BCB uses a public key for encryption or key-wrap. When using asymmetric keyed algorithms, the security source SHALL include a public key container or public key identifier as a signature or recipient header. The public key identifier SHALL be either an "x5t" or "x5chain" of [RFC9360], or "kid" (possibly with "kid context"), or an equivalent identifier.

When BIB result contains a "x5t" identifier, the security source MAY include an appropriate certificate container ("x5chain" or "x5bag") in a direct COSE header or an additional header security parameter (see Section 2.2.1). When a BIB result contains an "x5chain", the security source SHOULD NOT also include an "x5t" because the first certificate in the chain is implicitly the applicable end-entity certificate. For a BIB, if all potential security verifiers and acceptors are known to possess related public key and/or certificate data then the public key or additional header parameters can be omitted. Risks of not including related credential data are described in Section 5.5 and Section 5.6.

When present, public keys and certificates SHOULD be included as additional header parameters rather than within result COSE messages. This provides size efficiency when multiple security results are present because they will all be from the same security source and likely share the same public key material. Security verifiers and acceptors SHALL still process public keys or certificates present in a result message or recipient as applying to that individual COSE level.

Security verifiers and acceptors SHALL aggregate all COSE_Key objects from all parameters within a single BIB or BCB, independent of encoded type or order of parameters. Because each context contains a single set of security parameters which apply to all results in the same context, security verifiers and acceptors SHALL treat all public keys as being related to the security source itself and potentially applying to every result.

3.6. Policy Recommendations

The RECOMMENDED priority policy for including public key identifiers for BIB results is as follows:

1. When receivers are not known to possess certificate chains, a full chain is included (as an "x5chain").

2. When receivers are known to possess root and intermediate CAs, just the end-entity certificate is included (again as an "x5chain").

3. When receivers are known to possess associated chains including end-entity certificates, a certificate thumbnail (as an "x5t").

4. Some arbitrary identifier is used to correlate to an end-entity certificate (as a "kid" with an optional "kid context").

5. The BIB Security Source is used to imply an associated end-entity certificate which identifies that Node ID.

When certificates are used for public key data and the end-entity certificate is not explicitly trusted (i.e. pinned), a security verifier or acceptor SHALL perform the certification path validation of Section 2.6.1.2 up to one or more trusted CA certificates. Leaving out part of the certification chain can cause a security verifier or acceptor to fail to validate a BIB if the left-out certificates are unknown to the acceptor (see Section 5.6).

The RECOMMENDED priority policy for including public key identifiers for BCB results is as follows:

1. When receivers are known to possess associated end-entity certificates, a certificate thumbnail (as an "x5t").

2. Some arbitrary identifier is used to correlate to the private key (as a "kid" with an optional "kid context").

Any end-entity certificate associated with a BIB security source or BCB result recipient SHALL adhere to the profile of Section 4.

4. PKIX Certificate Profile

This section contains requirements on certificates used for the COSE context, while Section 3.5 contains requirements for how such certificates are transported or identified.

All end-entity X.509 certificates used for BPSec SHALL conform to [RFC5280], or any updates or successors to that profile.

This profile requires Version 3 certificates due to the extensions used by this profile. Security verifiers and acceptors SHALL reject as invalid Version 1 and Version 2 end-entity certificates.

Security verifiers and acceptors SHALL accept certificates that contain an empty Subject field or contain a Subject without a Common Name. Identity information in end-entity certificates SHALL be contained in the Subject Alternative Name extension in accordance with Section 4.2.1.6 of [RFC5280].

A BPSec end-entity certificate SHALL contain a NODE-ID in its Subject Alternative Name extension which authenticates the Node ID of the security source (for integrity) or a security verifier or acceptor (for confidentiality). The identifier type NODE-ID is defined in Section 4.4.1 of [RFC9174].

All end-entity and CA certificates used for BPSec SHOULD contain both a Subject Key Identifier extension in accordance with Section 4.2.1.2 of [RFC5280] and an Authority Key Identifier extension in accordance with Section 4.2.1.1 of [RFC5280]. Security verifiers and acceptors SHOULD NOT rely on either a Subject Key Identifier and an Authority Key Identifier being present in any received certificate. Including key identifiers simplifies the work of an entity needing to assemble a certification chain.

When allowed by CA policy, a BPSec end-entity certificate SHOULD contain a PKIX Extended Key Usage extension in accordance with Section 4.2.1.12 of [RFC5280]. When the PKIX Extended Key Usage extension is present, it SHALL contain a key purpose id-kp-bundleSecurity of [IANA-SMI]. The id-kp-bundleSecurity purpose MAY be combined with other purposes in the same certificate.

When allowed by CA policy, a BPSec end-entity certificate SHALL contain a PKIX Key Usage extension in accordance with Section 4.2.1.3 of [RFC5280]. The PKIX Key Usage bits which are consistent with COSE security are: digitalSignature, nonRepudiation, keyEncipherment, and keyAgreement. The specific algorithms used by COSE messages in security results determine which of those key uses are exercised. See Section 4.1 for discussion of key use policies across multiple certificates.

A BPSec end-entity certificate MAY contain an Online Certificate Status Protocol (OCSP) URI within an Authority Information Access extension in accordance with Section 4.2.2.1 of [RFC5280]. Security verifiers and acceptors are not expected to have continuous internet connectivity sufficient to perform OCSP verification.

4.1. Multiple-Certificate Uses

A RECOMMENDED security policy is to limit asymmetric keys (and thus public key certificates) to single uses among the following:

Bundle transport: With key uses as defined in the convergence layer specification(s). Transports can require additional Extended Key Usage, such as id-kp-serverAuth or id-kp-clientAuth.

Block signing: With key use digitalSignature and/or nonRepudiation.

Block encryption: With key use keyEncipherment and/or keyAgreement.

This policy is the same one recommended by Section 6 of [RFC8551] for email security and by Section 5.2 of [SP800-57] more generally. Effectively this means that a BP node uses separate certificates for transport (e.g., as a TCPCL entity), BIB signing (as a security source), and BCB encryption (as a security acceptor).

5. Security Considerations

This section separates security considerations into threat categories based on guidance of BCP 72 [RFC3552].

5.1. Threat: BPSec Block Replay

The bundle’s primary block contains fields which uniquely identify a bundle: the Source Node ID, Creation Timestamp, and fragment parameters (see Section 4.3.1 of [RFC9171]). These same fields are used to correlate Administrative Records with the bundles for which the records were generated. Including the primary block in the AAD Scope for integrity and confidentiality (see Section 2.2.2) binds the verification of the secured block to its parent bundle and disallows replay of any block with its BIB or BCB.

This profile of COSE limits the encryption algorithms to only AEAD in order to include the context of the encrypted data as AAD. If an agent mistakenly allows the use of non-AEAD encryption when decrypting and verifying a BCB, the possibility of block replay attack is present.

5.2. Threat: Untrusted End-Entity Certificate

The profile in Section 2.6.1 uses end-entity certificates chained up to a trusted root CA, where each certificate has a specific validity time interval.

A security verifier or acceptor needs to assemble an entire certificate chain in order to validate the use of an end-entity certificate. A security source can include a certificate set which does not contain the full chain, possibly excluding intermediate or root CAs. In an environment where security verifiers and acceptors are known to already contain needed root and intermediate CAs there is no need to include those CAs, but this has a risk of a relying node not actually having one of the needed CAs.

A security verifier or acceptor needs to use the bundle creation time when assembling a certificate chain and and validating it. Because of this, a security source needs to use the bundle creation time as the specific instant for choosing appropriate certificate(s) based on their validity time interval. The selection of a certificate outside of its validity time period will cause the entire security operation to be unusable.

5.3. Threat: Certificate Validation Vulnerabilities

Even when a security acceptor is operating properly an attacker can attempt to exploit vulnerabilities within certificate check algorithms or configuration to authenticate using an invalid certificate. An invalid certificate exploit could lead to higher-level security issues and/or denial of service to the Node ID being impersonated.

There are many reasons, described in [RFC5280] and [RFC6125], why a certificate can fail to validate, including using the certificate outside of its validity time interval, using purposes for which it was not authorized, or using it after it has been revoked by its CA. Validating a certificate is a complex task and can require network connectivity outside of the primary BP convergence layer network path(s) if a mechanism such as OCSP [RFC6960] is used by the CA. The configuration and use of particular certificate validation methods are outside of the scope of this document.

5.4. Threat: Security Source Impersonation

When certificates are referenced by BIB results it is possible that the certificate does not contain a NODE-ID or does contain one but has a mismatch with the actual security source (see Section 1.2). Having a CA-validated certificate does not alone guarantee the identity of the security source from which the certificate is provided; additional validation procedures in Section 2.6.1 bind the Node ID based on the contents of the certificate.

5.5. Threat: Unidentifiable Key

The profile in Section 3.2 recommends key identifiers when possible and the parameters in section Section 2.2 allow encoding public keys where available. If the application using a COSE Integrity or COSE Confidentiality context leaves out key identification data (in a COSE recipient structure), a security verifier or acceptor for those BPSec blocks only has the primary block available to use when verifying or decrypting the target block. This leads to a situation, identified in BPSec Security Considerations, where a signature is verified to be valid but not from the expected Security Source.

Because the key identifier headers are unprotected (see Section 3.5), there is still the possibility that an active attacker removes or alters key identifier(s) in the result. This can cause a security verifier or acceptor to not be able to properly verify a valid signature or not use the correct private key to decrypt valid ciphertext.

5.6. Threat: Non-Trusted Public Key

The profile in Section 3.2 allows the use of PKIX which typically involves end-entity certificates chained up to a trusted root CA. A BIB can reference or contain end-entity certificates not previously known to a security acceptor but the acceptor can still trust the certificate by verifying it up to a trusted CA. In an environment where security verifiers and acceptors are known to already contain needed root and intermediate CAs there is no need to include those CAs in a proper chain within the security parameters, but this has a risk of an acceptor not actually having one of the needed CAs.

Because the security parameters are not included as AAD, there is still the possibility that an active attacker removes or alters certification chain data in the parameters. This can cause a security verifier or acceptor to be able to verify a valid signature but not trust the public key used to perform the verification.

5.7. Threat: Passive Leak of Key Material

It is important that the key requirements of Section 2.2 apply only to public keys and PKIX certificates. Including non-public key material in ASB parameters will expose that material in the bundle data and over the bundle convergence layer during transport.

5.8. Threat: Algorithm Vulnerabilities

Because this use of COSE leaves the specific algorithms chosen for BIB and BCB use up to the applications securing bundle data, it is important to use only COSE algorithms which are marked as "recommended" in the IANA registry [IANA-COSE].

5.9. Inherited Security Considerations

All of the security considerations of the underlying BPSec [RFC9172] apply to this security context. Because this security context uses whole COSE messages and inherits all COSE processing, all of the security considerations of [RFC9052] apply to this security context.

5.10. AAD-Covered Block Modification

The AAD Scope parameter (Section 2.2.2) can be used to refer to any other block within the same bundle (by its unique block number) at the time the associated security operation is added to a bundle. Because of this, if any block within the AAD coverage is modified (by any node along the bundle’s forwarding path) in a way which affects the generated AAD value (Section 2.5.1) it will cause verification or acceptance of the containing security operation to fail.

One reason why such a modification would be made is that the other block has an expected lifetime shorter than the security operation. For example, a Previous Node block (Section 4.4.1 of [RFC9171]) is expected to be removed or replaced at each hop. The AAD Scope parameter SHALL NOT reference any other block with an expected lifetime shorter than the containing security operation.

Another reason for a modification is that the other block is designed to be updated along the forwarding path. For example, a Hop Count block (Section 4.4.3 of [RFC9171]) is expected to be modified as the bundle is forwarded by each node. The AAD Scope parameter SHALL NOT reference any other block using the flag AAD-btsd (Table 2) if that other block is expected to be modified by intermediate nodes during the lifetime of the containing security operation.

One reason for a block to be removed is if it has its block processing control flags (Section 4.2.4 of [RFC9171]) have the bit set indicating "Discard block if it can't be processed" and the block type or type-specific data cannot be handled by any node along the forwarding path. The AAD Scope parameter SHALL NOT reference any other block having block processing control flags with the bit set indicating "Discard block if it can't be processed" unless it is known that all possible receiving nodes can process the associated block type during the lifetime of the containing security operation.

6. IANA Considerations

Registration procedures referred to in this section are defined in [RFC8126].

6.1. Bundle Protocol

Within the "Bundle Protocol" registry group [IANA-BUNDLE], the following entry has been added to the "BPSec Security Context Identifiers" registry.

Value	Description	Reference
3	COSE	[This specification]

Table 7: BPSec Security Context Identifiers

Within the "Bundle Protocol" registry group [IANA-BUNDLE], the IANA has created and now maintains a new registry named "BPSec COSE AAD Scope Flags". Table 8 shows the initial values for this registry.

The registration policy for this registry is Specification Required.

The value range is unsigned 64-bit integer.

Bit Position (from LSbit)	Name	Reference
0	AAD-metadata	[This specification]
1	AAD-btsd	[This specification]
2-64	Unassigned	[This specification]

Table 8: BPSec COSE AAD Scope Flags

7. References

7.1. Normative References

[IANA-BUNDLE]
IANA, "Bundle Protocol",
<https://www.iana.org/assignments/bundle/>.

[IANA-COSE]
IANA, "CBOR Object Signing and Encryption (COSE)",
<https://www.iana.org/assignments/cose/>.

[IANA-SMI] IANA, "Structure of Management Information (SMI) Numbers",
<https://www.iana.org/assignments/smi-numbers/>.

[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>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

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

[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>.

[RFC8230] Jones, M., "Using RSA Algorithms with CBOR Object Signing and Encryption (COSE) Messages", RFC 8230, DOI 10.17487/RFC8230, September 2017, <https://www.rfc-editor.org/info/rfc8230>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

[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>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/info/rfc9053>.

[RFC9172] Birrane, III, E. and K. McKeever, "Bundle Protocol Security (BPSec)", RFC 9172, DOI 10.17487/RFC9172, January 2022, <https://www.rfc-editor.org/info/rfc9172>.

[RFC9174] Sipos, B., Demmer, M., Ott, J., and S. Perreault, "Delay-Tolerant Networking TCP Convergence-Layer Protocol Version 4", RFC 9174, DOI 10.17487/RFC9174, January 2022, <https://www.rfc-editor.org/info/rfc9174>.

[RFC9360] Schaad, J., "CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates", RFC 9360, DOI 10.17487/RFC9360, February 2023, <https://www.rfc-editor.org/info/rfc9360>.

7.2. Informative References

[SP800-57] US National Institute of Standards and Technology, "Recommendation for Key Management - Part 1: General", NIST SP 800-57, May 2020, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf>.

[FIPS-140] US National Institute of Standards and Technology, "Security Requirements for Cryptographic Modules", FIPS 140-3, March 2019, <https://doi.org/10.6028/NIST.FIPS.140-3>.

[CNSA1] US Committee on National Security Systems, "Use of Public Standards for Secure Information Sharing", CNSS Policy 15, 20 October 2016.

[CNSA2] US Committee on National Security Systems, "Use of Public Standards for Secure Information Sharing", CNSS Policy 15, December 2024.

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

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

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

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

[github-dtn-bpsec-cose] Sipos, B., "DTN Bundle Protocol Security COSE Security Context", <https://github.com/BrianSipos/dtn-bpsec-cose/>.

[github-dtn-demo-agent] Sipos, B., "Demo Convergence Layer Agent", <https://github.com/BrianSipos/dtn-demo-agent/>.

[gitlab-wireshark] Wireshark Foundation, "Wireshark repository", <https://gitlab.com/wireshark/wireshark>.

Appendix A. Example Security Operations

These examples are intended to have the correct structure of COSE security blocks but in some cases use simplified algorithm parameters or smaller key sizes than are required by the actual COSE profile defined in this documents. Each example indicates how it differs from the actual profile if there is a meaningful difference.

All of these examples operate within the context of the bundle primary block of Figure 6 with a security target block of Figure 7. All example figures use the extended diagnostic notation [RFC8610].

```
[
  7, / BP version /
  0, / flags /
  0, / CRC type /
  [1, "//dst/svc"], / destination /
  [1, "//src/svc"], / source /
  [1, "//src/"], / report-to /
  [ / timestamp: /
    813110400000, / creation time: 2025-10-07T00:00:00Z /
    0 / seq. no. /
  ],
  1000000 / lifetime /
]
```

Figure 6: Primary block CBOR diagnostic

```
[
  1, / type code: payload /
  1, / block num /
  0, / flags /
  0, / CRC type /
  <<"hello">> / block-type-specific-data /
]
```

Figure 7: Target block CBOR diagnostic

Together these form an original bundle without any security operations present. This bundle is encoded as the following 67 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850101000046656865 6c6c6fff
```

All of the block integrity block examples operate within the context of the "frame" block of Figure 8, and block confidentiality block examples within the frame block of Figure 9.

```
[
  11, / type code: BIB /
  3, / block num /
  0, / flags /
  0, / CRC type /
  '' / BTSD to be replaced with ASB /
]
```

Figure 8: Block integrity frame block CBOR diagnostic

```
[
  12, / type code: BCB /
  3, / block num /
  0, / flags /
  0, / CRC type /
  '' / BTSD to be replaced with ASB /
]
```

Figure 9: Block confidentiality frame block CBOR diagnostic

All of the examples also operate within a security block containing the AAD Scope parameter with value {0:0b1,-1:0b1} indicating the primary block and target block metadata are included. This results in a consistent AAD-list as shown in Figure 10, which is encoded as the byte string for COSE external_aad in all of the examples.

```
[1, "//src/"], / security source /
{0:0b1, -1:0b1}, / AAD-scope /
[7, 0, 0, [1, "//dst/svc"], [1, "//src/svc"], [1, "//src/"],
  [813110400000, 0 ], 1000000], / primary-block /
1, 1, 0, / target block-metadata /
'' / additional-protected /
```

Figure 10: Example scope AAD-list CBOR-sequence diagnostic

The only differences between these examples which use EC or RSA keypairs and a use of a public key certificate are: the highest-layer parameters would contain an "x5t" (or equivalent, see Section 3.5) value instead of a "kid" value. This would not be a change to a protected header so, given the same private key, there would be no change to the signature or wrapped-key data.

Because each of the COSE_Encrypt examples use the same CEK within the same AAD, the target ciphertext is also identical. The target block after application of the encryption is shown in Figure 11.

```
[
  1, / type code: payload /
  1, / block num /
  0, / flags /
  0, / CRC type /
  h'1fd25f64a2ee886e97ecfde7667371214f5add54a089' / ciphertext /
]
```

Figure 11: Encrypted Target block CBOR diagnostic

A.1. Symmetric Key COSE_Mac0

This is an example of a MAC with recipient having a 256-bit symmetric key identified by a "kid".

```
[
  {
    / kty / 1: 4, / symmetric /
    / kid / 2: 'ExampleMAC',
    / alg / 3: 5, / HMAC 256//256 /
    / ops / 4: [9, 10], / MAC create, MAC verify /
    / k / -1: h'13bf9cead057c0aca2c9e52471ca4b19ddfaf4c0784e3f3e8e39
99dbae4ce45c'
  }
]
```

Figure 12: Symmetric Key

The external_aad is the encoded data from Figure 10. The payload is the encoded target BTSD from Figure 7.

```
[
  "MAC0", / context /
  h'a10105', / protected /
  h'8201662f2f7372632fa200012001880700008201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
4001010040', / external_aad /
  h'6568656c6c6f' / payload /
]
```

Figure 13: MAC_structure CBOR diagnostic

```
[1], / targets /
3, / security context /
1, / flags: params-present /
[1, "//src/"], / security source /
[ / parameters /
  [
    5, / AAD-scope /
    {0:0b1,-1:0b1} / primary metadata, target metadata /
  ]
],
[
  [ / target block #1 /
    [ / result /
      17, / COSE_Mac0 tag /
      <<[
        <<{ / protected /
           / alg / 1: 5 / HMAC 256//256 /
        }>>,
        { / unprotected /
          / kid / 4: 'ExampleMAC'
        },
        null, / payload detached /
        h'c2554546bed1e5ed419aab9d4450fc62fb6b64ea454e53ff83eeb9f87c
052606' / tag /
      ]>>
    ]
  ]
]
```

Figure 14: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 154 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850b0300005850810103018201662
2f2f7372632f818205a200012001818182115835844 3a10105a1044a4578616d706c
654d4143f65820c2554546bed1e5ed419aab9d4450fc62fb6b64ea454e53ff83eeb9
f87c052606850101000046656 8656c6c6fff
```

A.2. EC Keypair COSE_Sign1

This is an example of a signature with a recipient having a P-256 curve EC keypair identified by a "kid". The associated public key is included as a security parameter.

```
[
  { / signing private key /
    / kty / 1: 2, / EC2 /
    / kid / 2: 'ExampleEC2',
    / alg / 3: -9, / ESP256 /
    / ops / 4: [1, 2], / sign, verify /
    / crv / -1: 1, / P-256 /
    / x / -2: h'44c1fa63b84f172b50541339c50beb0e630241ecb4eebbddb8b5
e4fe0a1787a8',
    / y / -3: h'059451c7630d95d0b550acbd02e979b3f4f74e645b74715fafbc
1639960a0c7a',
    / d / -4: h'dd6e7d8c4c0e0c0bd3ae1b4a2fa86b9a09b7efee4a233772cf51
89786ea63842'
  }
]
```

Figure 15: Example Keys

The external_aad is the encoded data from Figure 10. The payload is the encoded target BTSD from Figure 7.

```
[
  "Signature1", / context /
  h'a10126', / protected /
  h'8201662f2f7372632fa200012001880700008201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
4001010040', / external_aad /
  h'6568656c6c6f' / payload /
]
```

Figure 16: Sig_structure CBOR diagnostic

```
   [1], / targets /
   3, / security context /
   1, / flags: params-present /
   [1, "//src/"], / security source /
   [ / parameters /
     [
       5, / AAD-scope /
       {0:0b1,-1:0b1} / primary metadata, target metadata /
     ]
   ],
   [
     [ / target block #1 /
       [ / result /
         18, / COSE_Sign1 tag /
         <<[
           <<{ / protected /
              / alg / 1: -9 / ESP256 /
           }>>,
           { / unprotected /
             / kid / 4: 'ExampleEC2'
           },
           null, / payload detached /
           h'edf16f7619151b530e8ba675556644b07d5a0804ad5762d2e8260358aa
   ac221fd2984d03e8a854fd4da63f4e09324944d94367a5912fd7f38dae4362d67cfa
   c5' / signature /
         ]>>
       ]
     ]
   ]
```

Figure 17: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 186 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850b03000058708101030182016 6
2f2f7372632f818205a200012001818182125855844 3a10128a1044a4578616d706c
65454332f65840edf16f7619151b530e8ba675556644b07d5a0804ad5762d2e82603
58aaac221fd2984d03e8a854fd4da63f4e09324944d94367a5912fd7f38dae4362d6
7cfac5850101000046656 8656c6c6fff
```

A.3. RSA Keypair COSE_Sign1

This is an example of a signature with a recipient having a 1024-bit RSA keypair identified by a "kid". The associated public key is included as a security parameter.

This key strength is not supposed to be a secure configuration, only intended to explain the procedure. This signature uses a random salt, so the full signature output is not deterministic.

```
[
  { / signing private key /
    / kty / 1: 3, / RSA /
    / kid / 2: 'ExampleRSA',
    / alg / 3: -37, / PS256 /
    / ops / 4: [1, 2], / sign, verify /
    / n / -1: h'b0b5fd85f52c91844007443c9f9371980025f76d51fc9c676812
31da610cb291ba637ce813bffdb2e9c653258607389ec97dad3db295fded67744ed6
20707db36804e74e56a494030a73608fc8d92f2f0578d2d85cc201ef0ff22d7835d2
d147d3b90a6884276235a01c2be99dfc597f79554362fc1eb03639cac5ccaddb29
25',
    / e / -2: h'010001',
    / d / -3: h'9b5d26ad6445ef1aab80b809e4f329684e9912d556c4166f041d
1b1fb93c04b4037ffd0dbe6f8a8a86e70bab6e0f6344983a9ada27ed9ff7de816fde
eb5e7be48e607ce5fda4581ca6338a9e019fb3689b28934192b6a190cdda910abb5a
86a2f7b6f9cd5011049d8de52ddfef73aa06df401c55623ec196720f54920deb4f
01',
    / p / -4: h'db22d94e7784a27b568cbf985307ea8d6430ff6b88c18a7086fd
4f57a326572f2250c39e48a6f8e2201661c2dfe12c7386835b649714d050aa36123e
c3d00e75',
    / q / -5: h'ce7016adc5f326b7520397c5978ee2f50e69279983d54c5d76f0
5bcd61de0879d7056c923540dff9cbae95dcc0e5e86b52b3c902dc9669c8021c6955
7effb9f1',
    / dP / -6: h'6a6fcaccea106a3b2e16bf18e57b7ad9a2488a4758ed68a8af6
86a194f0d585b7477760c738d6665aee0302bcf4237ad0530d83b4b86b887f5a4bdc
7eea427e1',
    / dQ / -7: h'28a4cae245b1dcb285142e027a1768b9c4af915b59285a93a04
22c60e05edd9e57663afd023d169bd0ad3bd62da8563d231840802ebbf271ad70b89
05ba3af91',
    / qInv / -8: h'07b5a61733896270a6bd2bb1654194c54e2bc0e061b543a4e
d9fa73c4bc79c87148aa92a451c4ab8262b6377a9c7b97f869160ca6f5d853ee4b65
f4f92865ca3'
  }
]
```

Figure 18: Example Keys

The external_aad is the encoded data from Figure 10. The payload is the encoded target BTSD from Figure 7.

```
[
  "Signature1", / context /
  h'a1013824', / protected /
  h'8201662f2f7372632fa200012001880700008201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
4001010040', / external_aad /
  h'6568656c6c6f' / payload /
]
```

Figure 19: Sig_structure CBOR diagnostic

```
[1], / targets /
3, / security context /
1, / flags: params-present /
[1, "//src/"], / security source /
[ / parameters /
  [
    5, / AAD-scope /
    {0:0b1,-1:0b1} / primary metadata, target metadata /
  ]
],
[
  [ / target block #1 /
    [ / result /
      18, / COSE_Sign1 tag /
      <<[
        <<{ / protected /
           / alg / 1: -37 / PS256 /
        }>>,
        { / unprotected /
          / kid / 4: 'ExampleRSA'
        },
        null, / payload detached /
        h'863c06655b79b2c9b676d26ea5aa034faa4b765e6ca7516897a3dbf775
57cfa90f992264757f34b95f3934d736f808b2db1bdba96bc4b17daad8565f5b9652
44121dd5665c072723b8a3c94de8e287eb95d44d1481a5b1b63fc30237a1d4d87ece
0495a84393cc49ad377d907b7b38e85beafd5f3a18796809987d1341de034f'
 / signature /
      ]>>
    ]
  ]
]
```

Figure 20: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 251 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850b03000058b181010301820166
2f2f7372632f818205a200012001818182125896844 4a1013824a1044a4578616d70
6c65525341f65880863c06655b79b2c9b676d26ea5aa034faa4b765e6ca7516897a3
dbf77557cfa90f992264757f34b95f3934d736f808b2db1bdba96bc4b17daad8565f
5b965244121dd5665c072723b8a3c94de8e287eb95d44d1481a5b1b63fc30237a1d4
d87ece0495a84393cc49ad377d907b7b38e85beafd5f3a18796809987d1341de034f
850101000046656 8656c6c6fff
```

A.4. Symmetric CEK COSE_Encrypt0

This is an example of an encryption with an explicit CEK identified by a "kid". The key used is shown in Figure 21, which includes a Base IV parameter in order to reduce the total size of the COSE message using a Partial IV.

```
[
  {
    / kty / 1: 4, / symmetric /
    / kid / 2: 'ExampleCEK',
    / alg / 3: 3, / A256GCM /
    / ops / 4: [3, 4], / encrypt, decrypt /
    / base IV / 5: h'6f3093eba5d85143c3dc0000',
    / k / -1: h'13bf9cead057c0aca2c9e52471ca4b19ddfaf4c0784e3f3e8e39
99dbae4ce45c'
  }
]
```

Figure 21: Example Key

The external_aad is the encoded data from Figure 10.

```
[
  "Encrypt0", / context /
  h'a10103', / protected /
  h'8201662f2f7372632fa200012001880700008201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
4001010040' / external_aad /
]
```

Figure 22: Enc_structure CBOR diagnostic

The ASB item for this encryption operation is shown in Figure 23 and corresponds with the updated target block (containing the ciphertext) of Figure 24. This ciphertext is different than the common one in Figure 11 because of the different context string in Figure 22.

```
[1], / targets /
3, / security context /
1, / flags: params-present /
[1, "//src/"], / security source /
[ / parameters /
  [
    5, / AAD-scope /
    {0:0b1,-1:0b1} / primary metadata, target metadata /
  ]
],
[
  [ / target block #1 /
    [ / result /
      16, / COSE_Encrypt0 tag /
      <<[
        <<{ / protected /
           / alg / 1: 3 / A256GCM /
        }>>,
        { / unprotected /
          / kid / 4: 'ExampleCEK',
          / partial iv / 6: h'484a'
        },
        null / payload detached /
      ]>>
    ]
  ]
]
```

Figure 23: Abstract Security Block CBOR diagnostic

```
[
  1, / type code: payload /
  1, / block num /
  0, / flags /
  0, / CRC type /
  h'1fd25f64a2ee5c93bb884d529bce14cb24bdeaf8a3f1' / ciphertext /
]
```

Figure 24: Encrypted Target block CBOR diagnostic

The final bundle is encoded as the following 139 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850c0300005831810103018201662
2f2f7372632f818205a200012001818182105783 43a10103a2044a4578616d706c65
43454b0642484af68501010000561fd25f64a2ee5c93bb884d529bce14cb24bdeaf8
a3f1ff
```

A.5. Symmetric KEK COSE_Encrypt

This is an example of an encryption with a random CEK and an explicit key-encryption key (KEK) identified by a "kid". The keys used are shown in Figure 25.

```
[
  {
    / kty / 1: 4, / symmetric /
    / kid / 2: 'ExampleKEK',
    / alg / 3: -5, / A256KW /
    / ops / 4: [5, 6], / wrap, unwrap /
    / k / -1: h'0e8a982b921d1086241798032fedc1f883eab72e4e43bb2d11cf
ae38ad7a972e'
  },
  {
    / kty / 1: 4, / symmetric /
    / kid / 2: 'ExampleCEK',
    / alg / 3: 3, / A256GCM /
    / ops / 4: [3, 4], / encrypt, decrypt /
    / k / -1: h'13bf9cead057c0aca2c9e52471ca4b19ddfaf4c0784e3f3e8e39
99dbae4ce45c'
  }
]
```

Figure 25: Example Keys

The external_aad is the encoded data from Figure 10.

```
[
  "Encrypt", / context /
  h'a10103', / protected /
  h'8201662f2f7372632fa200012001880700008201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
4001010040' / external_aad /
]
```

Figure 26: Enc_structure CBOR diagnostic

The ASB item for this encryption operation is shown in Figure 27 and corresponds with the updated target block (containing the ciphertext) of Figure 11. The recipient does not have any protected header parameters because AES Key Wrap does not allow any AAD.

```
   [1], / targets /
   3, / security context /
   1, / flags: params-present /
   [1, "//src/"], / security source /
   [ / parameters /
     [
       5, / AAD-scope /
       {0:0b1,-1:0b1} / primary metadata, target metadata /
     ]
   ],
   [
     [ / target block #1 /
       [ / result /
         96, / COSE_Encrypt tag /
         <<[
           <<{ / protected /
              / alg / 1: 3 / A256GCM /
           }>>,
           { / unprotected /
             / iv / 5: h'6f3093eba5d85143c3dc484a'
           },
           null, / payload detached /
           [
             [ / recipient /
               h'', / protected /
               { / unprotected /
                 / alg / 1: -5, / A256KW /
                 / kid / 4: 'ExampleKEK'
               },
               h'917f2045e1169502756252bf119a94cdac6a9d8944245b5a9a26d4
   03a6331159e3d691a708e9984d' / key-wrapped /
             ]
           ]
         ]>>
       ]
     ]
   ]
```

Figure 27: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 199 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850c030000586d81010301820166
2f2f7372632f818205a200012001818182186058518443a10103a1054c6f3093eba5
d85143c3dc484af6818340a20124044a4578616d706c654b454b5828917f2045e116
9502756252bf119a94cdac6a9d8944245b5a9a26d403a6331159e3d691a708e9984d
8501010000561fd25f64a2ee886e97ecfde7667371214f5add54a089ff
```

A.6. EC Keypair COSE_Encrypt

This is an example of an encryption with an P-256 curve ephemeral sender keypair and a static recipient keypair identified by a "kid". The keys used are shown in Figure 28.

```
[
  { / sender ephemeral private key /
    / kty / 1: 2, / EC2 /
    / crv / -1: 1, / P-256 /
    / x / -2: h'fedaba748882050d1bef8ba992911898f554450952070aeb4788
ca57d1df6bcc',
    / y / -3: h'ceaa8e7ff4751a4f81c70e98f1713378b0bd82a1414a2f493c1c
9c0670f28d62',
    / d / -4: h'a2e4ed4f2e21842999b0e9ebdaad7465efd5c29bd5761f5c2088
0f9d9c3b122a'
  },
  { / recipient private key /
    / kty / 1: 2, / EC2 /
    / kid / 2: 'ExampleEC2',
    / alg / 3: -31, / ECDH-ES + A256KW /
    / ops / 4: [7], / derive key /
    / crv / -1: 1, / P-256 /
    / x / -2: h'44c1fa63b84f172b50541339c50beb0e630241ecb4eebbddb8b5
e4fe0a1787a8',
    / y / -3: h'059451c7630d95d0b550acbd02e979b3f4f74e645b74715fafbc
1639960a0c7a',
    / d / -4: h'dd6e7d8c4c0e0c0bd3ae1b4a2fa86b9a09b7efee4a233772cf51
89786ea63842'
  },
  {
    / kty / 1: 4, / symmetric /
    / kid / 2: 'ExampleCEK',
    / alg / 3: 3, / A256GCM /
    / ops / 4: [3, 4], / encrypt, decrypt /
    / k / -1: h'13bf9cead057c0aca2c9e52471ca4b19ddfaf4c0784e3f3e8e39
99dbae4ce45c'
  }
]
```

Figure 28: Example Keys

The external_aad is the encoded data from Figure 10.

```
[
  "Encrypt", / context /
  h'a10103', / protected /
  h'8201662f2f7372632fa200012001880700008201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
4001010040' / external_aad /
]
```

Figure 29: Enc_structure CBOR diagnostic

The ASB item for this encryption operation is shown in Figure 30 and corresponds with the updated target block (containing the ciphertext) of Figure 11.

```
   [1], / targets /
   3, / security context /
   1, / flags: params-present /
   [1, "//src/"], / security source /
   [ / parameters /
     [
       5, / AAD-scope /
       {0:0b1,-1:0b1} / primary metadata, target metadata /
     ]
   ],
   [
     [ / target block #1 /
       [ / result /
         96, / COSE_Encrypt tag /
         <<[
           <<{ / protected /
              / alg / 1: 3 / A256GCM /
           }>>,
           { / unprotected /
             / iv / 5: h'6f3093eba5d85143c3dc484a'
           },
           null, / payload detached /
           [
             [ / recipient /
               <<{ / protected /
                 / alg / 1: -31 / ECDH-ES + A256KW /
               }>>,
               { / unprotected /
                 / kid / 4: 'ExampleEC2',
                 / ephemeral key / -1: {
                   1: 2,
                   -1: 1,
                   -2: h'fedaba748882050d1bef8ba992911898f554450952070ae
   b4788ca57d1df6bcc',
                   -3: h'ceaa8e7ff4751a4f81c70e98f1713378b0bd82a1414a2f4
   93c1c9c0670f28d62'
                 }
               },
               h'a5ed54991baa90af6aa7f272e7fd456fc0ba36a507842aa1d0a577
   65386ab04d0fba5c454703366a' / key-wrapped /
             ]
           ]
         ]>>
       ]
     ]
   ]
```

Figure 30: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 277 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850c03000058bb81010301820166
2f2f7372632f818205a200012001818181821860589f8443a10103a1054c6f3093eba5
d85143c3dc484af6818344a101381ea2044a4578616d706c6545433220a401022001
215820fedaba748882050d1bef8ba992911898f554450952070aeb4788ca57d1df6b
cc225820ceaa8e7ff4751a4f81c70e98f1713378b0bd82a1414a2f493c1c9c0670f2
8d625828a5ed54991baa90af6aa7f272e7fd456fc0ba36a507842aa1d0a57765386a
b04d0fba5c454703366a8501010000561fd25f64a2ee886e97ecfde7667371214f5a
dd54a089ff
```

A.7. RSA Keypair COSE_Encrypt

This is an example of an encryption with a recipient having a 1024-bit RSA keypair identified by a "kid". The associated public key is included as a security parameter.

This key strength is not supposed to be a secure configuration, only intended to explain the procedure. This padding scheme uses a random salt, so the full Layer 1 ciphertext output is not deterministic.

```
[
  { / recipient private key /
    / kty / 1: 3, / RSA /
    / kid / 2: 'ExampleRSA',
    / alg / 3: -41, / RSAES-OAEP w SHA-256 /
    / ops / 4: [1, 2], / sign, verify /
    / n / -1: h'b0b5fd85f52c91844007443c9f9371980025f76d51fc9c676812
31da610cb291ba637ce813bffdb2e9c653258607389ec97dad3db295fded67744ed6
20707db36804e74e56a494030a73608fc8d92f2f0578d2d85cc201ef0ff22d7835d2
d147d3b90a6884276235a01c2be99dfc597f79554362fc1eb03639cac5ccaddb29
25',
    / e / -2: h'010001',
    / d / -3: h'9b5d26ad6445ef1aab80b809e4f329684e9912d556c4166f041d
1b1fb93c04b4037ffd0dbe6f8a8a86e70bab6e0f6344983a9ada27ed9ff7de816fde
eb5e7be48e607ce5fda4581ca6338a9e019fb3689b28934192b6a190cdda910abb5a
86a2f7b6f9cd5011049d8de52ddfef73aa06df401c55623ec196720f54920deb4f
01',
    / p / -4: h'db22d94e7784a27b568cbf985307ea8d6430ff6b88c18a7086fd
4f57a326572f2250c39e48a6f8e2201661c2dfe12c7386835b649714d050aa36123e
c3d00e75',
    / q / -5: h'ce7016adc5f326b7520397c5978ee2f50e69279983d54c5d76f0
5bcd61de0879d7056c923540dff9cbae95dcc0e5e86b52b3c902dc9669c8021c6955
7effb9f1',
    / dP / -6: h'6a6fcaccea106a3b2e16bf18e57b7ad9a2488a4758ed68a8af6
86a194f0d585b7477760c738d6665aee0302bcf4237ad0530d83b4b86b887f5a4bdc
7eea427e1',
    / dQ / -7: h'28a4cae245b1dcb285142e027a1768b9c4af915b59285a93a04
22c60e05edd9e57663afd023d169bd0ad3bd62da8563d231840802ebbf271ad70b89
05ba3af91',
    / qInv / -8: h'07b5a61733896270a6bd2bb1654194c54e2bc0e061b543a4e
d9fa73c4bc79c87148aa92a451c4ab8262b6377a9c7b97f869160ca6f5d853ee4b65
f4f92865ca3'
  },
  {
    / kty / 1: 4, / symmetric /
    / kid / 2: 'ExampleCEK',
    / alg / 3: 3, / A256GCM /
    / ops / 4: [3, 4], / encrypt, decrypt /
    / k / -1: h'13bf9cead057c0aca2c9e52471ca4b19ddfaf4c0784e3f3e8e39
99dbae4ce45c'
  }
]
```

Figure 31: Example Keys

The external_aad is the encoded data from Figure 10.

```
[
  "Encrypt", / context /
  h'a10103', / protected /
  h'8201662f2f7372632fa200012001880700008201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
4001010040' / external_aad /
]
```

Figure 32: Enc_structure CBOR diagnostic

The ASB item for this encryption operation is shown in Figure 33 and corresponds with the updated target block (containing the ciphertext) of Figure 11. The recipient does not have any protected header parameters because RSA OAEP does not allow any AAD.

```
   [1], / targets /
   3, / security context /
   1, / flags: params-present /
   [1, "//src/"], / security source /
   [ / parameters /
     [
       5, / AAD-scope /
       {0:0b1,-1:0b1} / primary metadata, target metadata /
     ]
   ],
   [
     [ / target block #1 /
       [ / result /
         96, / COSE_Encrypt tag /
         <<[
           <<{ / protected /
              / alg / 1: 3 / A256GCM /
           }>>,
           { / unprotected /
             / iv / 5: h'6f3093eba5d85143c3dc484a'
           },
           null, / payload detached /
           [
             [ / recipient /
               h'', / protected /
               { / unprotected /
                 / alg / 1: -41, / RSAES-OAEP w SHA-256 /
                 / kid / 4: 'ExampleRSA'
               },
               h'ac5d970e42df409ea9f5d5cd714287fb67284f8c19c691cad2993c
   7418f1605e0369322b5ce34eb676cc19ccb7ff3db97af8a4057d458f0a1d7b21ee8d
   55eaadb26fb2cba54e77b18e9563569bd625f082885540b07577d335e896f0cdcb66
   012b01ef38631006c1761719705a4e2ada4b8f963e78be67a07b8c30bfd4adb2fa'
    / key-wrapped /
             ]
           ]
         ]>>
       ]
     ]
   ]
```

Figure 33: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 288 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850c03000058c681010301820166
2f2f7372632f818205a200012001818182186058aa8443a10103a1054c6f3093eba5
d85143c3dc484af6818340a2013828044a4578616d706c655253415880ac5d970e42
df409ea9f5d5cd714287fb67284f8c19c691cad2993c7418f1605e0369322b5ce34e
b676cc19ccb7ff3db97af8a4057d458f0a1d7b21ee8d55eaadb26fb2cba54e77b18e
9563569bd625f082885540b07577d335e896f0cdcb66012b01ef38631006c1761719
705a4e2ada4b8f963e78be67a07b8c30bfd4adb2fa8501010000561fd25f64a2ee88
6e97ecfde7667371214f5add54a089ff
```

Appendix B. Example Public Key Certificates

This section contains example public key certificates corresponding to end-entity private keys and identities used in examples of Appendix A with structure and extensions conforming to the profile of Section 4. All of the example certificates contain a validity time interval extending a short amount around the original bundle creation time of the original bundle (Figure 6).

B.1. Root CA Certificate

This root CA certificate and private key are included for completeness in testing path validation (Section 2.6.1.2) with a full chain. This root CA does not allow any intermediates purely as an example, while a typical deployed PKI would separate a root CA from intermediate signing CA(s). It also does not include any Certificate Policies, Name Constraints, or Policy Constraints extensions as an operational CA might do to express or control how its subordinates are validated and used. It does, however, include an Extended Key Usage (EKU) value id-kp-bundleSecurity which indicates that this certificate tree is authorized for securing BP data.

```
Version: 3 (0x2)
Serial Number:
    15:15:ff:a7:40:a4:bd:73:f5:ba
Signature Algorithm: ecdsa-with-SHA256
Issuer: CN = Certificate Authority
Validity
    Not Before: Oct  6 00:00:00 2025 GMT
    Not After : Oct 16 00:00:00 2025 GMT
Subject: CN = Certificate Authority
Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        pub:
            04:be:8f:38:92:c4:7a:7d:e8:1a:78:e9:89:99:8f:
            ab:e3:c2:02:31:46:d8:0f:20:84:ee:7d:c3:6b:66:
            1b:ad:15:65:74:51:5c:ad:1a:aa:c9:0a:08:f3:a8:
            86:0c:8f:50:71:76:0c:b8:1f:1a:fa:d6:f2:db:60:
            9d:ce:c7:68:82
        ASN1 OID: prime256v1
        NIST CURVE: P-256
X509v3 extensions:
    X509v3 Basic Constraints: critical
        CA:TRUE, pathlen:0
    X509v3 Key Usage: critical
        Certificate Sign, CRL Sign
    X509v3 Extended Key Usage:
        1.3.6.1.5.5.7.3.35
    X509v3 Subject Key Identifier:
        B8:68:0D:30:94:6A:99:20:E3:43:56:12:AA:05:03:10:E4:CD:AC:90
    X509v3 Authority Key Identifier:
        B8:68:0D:30:94:6A:99:20:E3:43:56:12:AA:05:03:10:E4:CD:AC:90
```

Figure 34: CA Certificate Content

```
-----BEGIN CERTIFICATE-----
MIIBsjCCAVmgAwIBAgIKFRX/p0CkvXP1ujAKBggqhkjOPQQDAjAgMR4wHAYDVQQD
DBVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwHhcNMjUxMDA2MDAwMDAwWhcNMjUxMDE2
MDAwMDAwWjAgMR4wHAYDVQQDDBVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwWTATBgcq
hkjOPQIBBggqhkjOPQMBBwNCAAS+jziSxHp96Bp46YmZj6vjwgIxRtgPIITufcNr
ZhutFWV0UVytGqrJCgjzqIYMj1Bxdgy4Hxr61vLbYJ3Ox2iCo3sweTASBgNVHRMB
Af8ECDAGAQH/AgEAMA4GA1UdDwEB/wQEAwIBBjATBgNVHSUEDDAKBggrBgEFBQcD
IzAdBgNVHQ4EFgQUuGgNMJRqmSDjQ1YSqgUDEOTNrJAwHwYDVR0jBBgwFoAUuGgN
MJRqmSDjQ1YSqgUDEOTNrJAwCgYIKoZIzj0EAwIDRwAwRAIgBx1VmoLrEcBWij6y
b4JPdOB198Mr4qQc6qzrvOjLh9kCIGolIBnHp57MW3RNcyTVEcqaSgK7H/UPB2f0
072iKC7G
-----END CERTIFICATE-----
```

Figure 35: CA Certificate PEM

```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIMvwM7fudPlsFOJDJGuG2HAlfg+MaFFisxBkicaPrYLBoAoGCCqGSM49
AwEHoUQDQgAEvo84ksR6fegaeOmJmY+r48ICMUbYDyCE7n3Da2YbrRVldFFcrRqq
yQoI86iGDI9QcXYMuB8a+tby22Cdzsdogg==
-----END EC PRIVATE KEY-----
```

Figure 36: CA Private Key PEM

B.2. Signing Source End-Entity Certificate

This end-entity certificate corresponds with the private key used for signing in Appendix A.2. It contains a SAN authenticating the single security source from that example, an EKU authorizing the identity, and a Key Usage authorizing the signing.

```
Version: 3 (0x2)
Serial Number:
    6f:fe:89:dc:b7:6e:d3:72:ea:7a
Signature Algorithm: ecdsa-with-SHA256
Issuer: CN = Certificate Authority
Validity
    Not Before: Oct  6 00:00:00 2025 GMT
    Not After : Oct 16 00:00:00 2025 GMT
Subject: CN = src
Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        pub:
            04:44:c1:fa:63:b8:4f:17:2b:50:54:13:39:c5:0b:
            eb:0e:63:02:41:ec:b4:ee:bb:dd:b8:b5:e4:fe:0a:
            17:87:a8:05:94:51:c7:63:0d:95:d0:b5:50:ac:bd:
            02:e9:79:b3:f4:f7:4e:64:5b:74:71:5f:af:bc:16:
            39:96:0a:0c:7a
        ASN1 OID: prime256v1
        NIST CURVE: P-256
X509v3 extensions:
    X509v3 Basic Constraints: critical
        CA:FALSE
    X509v3 Subject Alternative Name:
        othername: 1.3.6.1.5.5.7.8.11::dtn://src/
    X509v3 Key Usage: critical
        Digital Signature
    X509v3 Extended Key Usage:
        1.3.6.1.5.5.7.3.35
    X509v3 Subject Key Identifier:
        14:C1:F4:6F:29:03:0D:06:1D:9F:00:B5:52:81:B9:04:5D:DC:99:F0
    X509v3 Authority Key Identifier:
        B8:68:0D:30:94:6A:99:20:E3:43:56:12:AA:05:03:10:E4:CD:AC:90
```

Figure 37: Signing Certificate Content

```
-----BEGIN CERTIFICATE-----
MIIBwjCCAWigAwIBAgIKb/6J3Ldu03LqejAKBggqhkjOPQQDAjAgMR4wHAYDVQQD
DBVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwHhcNMjUxMDA2MDAwMDAwWhcNMjUxMDE2
MDAwMDAwWjAOMQwwCgYDVQQDDANzcmMwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNC
AAREwfpjuE8XK1BUEznFC+sOYwJB7LTuu924teT+CheHqAWUUcdjDZXQtVCsvQLp
ebP0905kW3RxX6+8FjmWCgx6o4GbMIGYMAwGA1UdEwEB/wQCMAAwIwYDVR0RBBww
GqAYBggrBgEFBQcIC6AMFgpkdG46Ly9zcmMvMA4GA1UdDwEB/wQEAwIHgDATBgNV
HSUEDDAKBggrBgEFBQcDIzAdBgNVHQ4EFgQUFMH0bykDDQYdnwC1UoG5BF3cmfAw
HwYDVR0jBBgwFoAUuGgNMJRqmSDjQ1YSqgUDEOTNrJAwCgYIKoZIzj0EAwIDSAAw
RQIhAKQf6kf0WhUxbFPEX3xdtdrnEKj9rssF/zdKPisygW7zAiBspYku9G6LrStY
hZ50V8OpuCMOz800PMREjFgSwQga5g==
-----END CERTIFICATE-----
```

Figure 38: Signing Certificate PEM

B.3. Encryption Recipient End-Entity Certificate

This end-entity certificate corresponds with the private key used for decrypting Appendix A.6. It contains a SAN identifying the single security acceptor from that example, an EKU authorizing the identity, and a Key Usage authorizing the key agreement.

```
Version: 3 (0x2)
Serial Number:
    3f:24:0b:cd:a6:f7:fc:3c:29:de
Signature Algorithm: ecdsa-with-SHA256
Issuer: CN = Certificate Authority
Validity
    Not Before: Oct  6 00:00:00 2025 GMT
    Not After : Oct 16 00:00:00 2025 GMT
Subject: CN = dst
Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        pub:
            04:47:f4:03:8f:0a:30:62:81:d6:14:c6:73:c8:a3:
            7b:88:d9:04:2d:75:77:67:51:a6:8c:dd:69:e1:a6:
            5e:25:54:38:01:0c:21:85:55:d4:31:e8:07:8f:9a:
            b8:75:9e:06:b9:bd:17:f5:e2:7e:0b:28:0a:e5:e6:
            15:e5:9f:a0:bd
        ASN1 OID: prime256v1
        NIST CURVE: P-256
X509v3 extensions:
    X509v3 Basic Constraints: critical
        CA:FALSE
    X509v3 Subject Alternative Name:
        othername: 1.3.6.1.5.5.7.8.11::dtn://dst/
    X509v3 Key Usage: critical
        Key Agreement
    X509v3 Extended Key Usage:
        1.3.6.1.5.5.7.3.35
    X509v3 Subject Key Identifier:
        84:26:AB:FD:B7:9D:BF:EC:CB:2A:80:78:BA:1B:23:36:E4:4B:49:FE
    X509v3 Authority Key Identifier:
        B8:68:0D:30:94:6A:99:20:E3:43:56:12:AA:05:03:10:E4:CD:AC:90
```

Figure 39: Key-Agreement Certificate Content

```
-----BEGIN CERTIFICATE-----
MIIBwzCCAWigAwIBAgIKPyQLzab3/Dwp3jAKBggqhkjOPQQDAjAgMR4wHAYDVQQD
DBVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwHhcNMjUxMDA2MDAwMDAwWhcNMjUxMDE2
MDAwMDAwWjAOMQwwCgYDVQQDDANkc3QwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNC
AARH9AOPCjBigdYUxnPIo3uI2QQtdXdnUaaM3Wnhpl4lVDgBDCGFVdQx6AePmrh1
nga5vRf14n4LKArl5hXln6C9o4GbMIGYMAwGA1UdEwEB/wQCMAAwIwYDVR0RBBww
GqAYBggrBgEFBQcIC6AMFgpkdG46Ly9kc3QvMA4GA1UdDwEB/wQEAwIDCDATBgNV
HSUEDDAKBggrBgEFBQcDIzAdBgNVHQ4EFgQUhCar/bedv+zLKoB4uhsjNuRLSf4w
HwYDVR0jBBgwFoAUuGgNMJRqmSDjQ1YSqgUDEOTNrJAwCgYIKoZIzj0EAwIDSQAw
RgIhAM9au3ladbH2Fjowc5NadsJR8EqNYOiFBwggtlAl/1tUAiEAhtJ7BUCa8n9h
F3x71sqspvbkQjp2SoLSA3loW73K2jw=
-----END CERTIFICATE-----
```

Figure 40: Key-Agreement Certificate PEM

Acknowledgments

Thanks to Lars Baumgaertner and Lukas Holst at ESA for review and prototyping feedback.

The interoperability minimum algorithms and their parameters are based on those available in the Default Security Contexts of [RFC9173].

Implementation Status

This section is to be removed before publishing as an RFC.

[NOTE to the RFC Editor: please remove this section before publication, as well as the reference to [RFC7942], [github-dtn-bpsec-cose], [github-dtn-demo-agent], and [gitlab-wireshark].]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations can exist.

A limited implementation of this COSE Context has been added to the [github-dtn-demo-agent] to help with interoperability testing.

As of the time of writing a COSE Context dissector has been accepted to the default development branch of the Wireshark project [gitlab-wireshark]. That dissector integrates the full-featured COSE dissector on top of BPSec, so will scale with any future additions to COSE itself.

An example implementation of this COSE Context has been created as a GitHub project [github-dtn-bpsec-cose] and is intended to use as a proof-of-concept and as a source of data for the examples in Appendix A. This example implementation only handles CBOR encoding/decoding and cryptographic functions, it does not construct actual BIB or BCB and does not integrate with a BP Agent.

Author’s Address

Brian Sipos
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
United States of America
Email: brian.sipos+ietf@gmail.com