Delay-Tolerant Networking
Internet-Draft
Updates: 9172 (if approved)
Intended status: Standards Track
Expires: 16 May 2026

B. Sipos
JHU/APL

12 November 2025

Bundle Protocol Security (BPSec) COSE Context draft-ietf-dtn-bpsec-cose-12

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 16 May 2026.

Table of Contents

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 rules of this document's CDDL, including the ASB data structure with its parameter/result sockets and rules needed for validating the example CBOR content.

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

The definitions for the rules MAC_structure, Sig_structure, Enc_structure, and COSE_KeySet are taken from COSE [RFC9052]. The definition for the rule COSE_CertHash is taken from COSE X.509 [RFC9360]. The definitions for the rules eid, primary-block, and extension-block, the sockets $extension-block, and the generic rule extension-block-use are taken from BP [RFC9171]. The BPSec specification [RFC9172] did not define its extension blocks using CDDL, so a supplementary definition for BPSec is provided in Appendix C.

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

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 uint or nint items) and values representing a collection of bit flags (as uint items) as defined below.

Non-negative integer AAD Scope keys SHALL be interpreted as block numbers in the bundle containing the AAD Scope parameter. Negative integer AAD Scope keys SHALL be interpreted as special (non-block-number) identifiers according to the IANA registry defined in Section 6.1. That registry contains the following initial values from Table 2 as well as reserved blocks for experimental and private use.

Value	Name	Description
-1	Target block	Include the target block of the security operation associated with the AAD.
-2	Security block	Include the security block containing the security operation associated with the AAD.

Table 2: AAD Scope Special Keys

AAD Scope values SHALL be interpreted as bit flags according to the IANA registry defined in Section 6.1 with initial values defined in Table 3. 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.

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 3: AAD Scope Flags

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

```
$bpsec-cose-param /= [5, AAD-scope]
AAD-scope = {
    *AAD-scope-key => AAD-scope-val
}
; All uint keys are block numbers
AAD-scope-key = uint / ($blk-id-special .within (-1 .. -65536))
$blk-id-special /= -1 ; target block
$blk-id-special /= -2 ; security block

AAD-scope-val = uint .bits $AAD-scope-flags
$AAD-scope-flags /= 0 ; AAD-metadata
$AAD-scope-flags /= 1 ; AAD-btsd
```

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 4. Each integrity result value SHALL consist of the COSE message indicated by Table 4 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 4: 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 one of the following: the encoded form of the primary block if the target is the primary block (block number zero), or the BTSD content of the target if the target is not the primary block (block number non-zero).

2.3.2. Confidentiality Messages

When used within a Block Confidentiality Block, COSE context SHALL allow only the Result IDs from Table 5. Each confidentiality result value SHALL consist of the COSE message indicated by Table 5 in its non-tagged encoded form.

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

Table 5: 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 content 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.

If the AAD Scope map contains any key which is a positive integer (block number) referencing a block which does not exist in the current bundle or any key which is a negative integer (special key) not supported by the processing entity the generation of the AAD SHALL be considered failed.

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 special keys 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) layer in accordance with [RFC9052]. When available, each COSE message SHALL contain a key identifier in the last layer for all signatures or recipients. See Section 3.4 and Section 3.5 for specifics about key identifiers. When a key identifier is not available, BPSec verifiers and acceptors SHALL use the Security Source 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 support using shared symmetric keys using modern key strengths, with recommended algorithms to support elliptic curve cryptography (ECC) 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 minimum set of COSE algorithms needed for interoperability in non-constrained devices is listed in this section and organized by the type of associated key material. This profile intentionally does not prohibit the use of any other algorithms in specific implementations, devices, or networks and is meant only to provide a starting point for general purpose implementations. It also does not address post-quantum algorithms which have been finalized by NIST but are still undergoing standardization in the IETF (see Section 5.8. The full set of COSE algorithms available is managed at [IANA-COSE].

Each algorithm in this profile is marked as being US CNSS CNSA 1.0 conformant [CNSA1] or CNSA 2.0 conformant [CNSA2] to aid in further narrowing of network-specific profiles and implementations. All of these algorithms in this profile are approved by US NIST FIPS 140-3 [FIPS-140], however FIPS 140 certification involves review of software and hardware design and implementation detail outside the scope of this document.

The threshold for minimum security strength to be included in this interoperability minimum is roughly equivalent to CNSA 1.0 and the CCSDS Space Data Link Security rationale green book [SDLS]. The breadth of algorithm variety is intended to cover many different current use cases beyond simple symmetric key security and be compatible with current PKIX mechanisms and strategies.

3.2.1. Hashing Algorithms

Implementations conforming to this specification SHALL support the non-keyed hash algorithms in Table 6 if they will operate with public key certificates.

Name	Code	Conformance
SHA-256/64	-15	
SHA-256	-16	
SHA-512/256	-17	
SHA-384	-43	CNSA 1.0 and 2.0
SHA-512	-44	CNSA 2.0

Table 6: Hash Algorithms

These algorithms are currently used in the COSE_CertHash of "x5t" header parameters, which are expected to be included as unprotected (see Section 3.5). The truncated algorithms are useful for certificate filtering using shorter thumbprints, so are included here even though they fall below the CNSA 1.0 minimum strength for protecting data.

3.2.2. Symmetric Algorithms

Implementations conforming to this specification SHALL support the symmetric keyed algorithms in Table 7.

> | The symmetric keyed algorithms here are not a super-set of those available in [RFC9173], this list includes only those which are CNSA 1.0 or 2.0 conformant.

The "direct" algorithm is really just a recipient placeholder to allow using a content encryption key (CEK) identifier in a that COSE layer, so has no cryptographic function or effect on security strength.

BPSec Block	COSE Layer	Name	Code	Conformance
Integrity	0	HMAC 384/384	6	CNSA 1.0 and 2.0
Integrity	0	HMAC 512/512	7	CNSA 2.0
Confidentiality	0	A256GCM	3	CNSA 1.0 and 2.0
Integrity or Confidentiality	1	A256KW	-5	CNSA 1.0 and 2.0
Integrity or Confidentiality	1	direct	-6	_N/A_
Integrity or Confidentiality	1	direct+HKDF-SHA-512	-11	CNSA 1.0 and 2.0

Table 7: Symmetric Algorithms

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).

The key length used for HMAC algorithms SHALL be equal to the hash function output length. This is consistent with COSE requirements on derived keys for HMAC but more strict to apply to all keys used for HMAC.

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.

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) CEK. When a COSE_Encrypt is used with an overall KEK, the recipient layer SHALL include a key identifier for the KEK.

When a COSE_Encrypt is used with a symmetric KEK and a single recipient, the direct HKDF algorithms (code -10 and -11) are RECOMMENDED over the key wrapped algorithms (code -3 through -5) to reduce message size and the need for symmetric key generation. When processing a COSE_Encrypt with a symmetric KEK, a security verifier or acceptor SHALL process all KDF 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.

3.2.3. ECC Algorithms

Implementations conforming to this specification SHALL support the ECC algorithms in Table 8 if they will operate with ECC key material using NIST curves.

| The ECC-based algorithms are CNSA 1.0 conformant [CNSA1] only
| when used with a key having curve P-384.

BPSec Block	COSE Layer	Name	Code	Conformance
Integrity	0 or 1	ESP384	-51	CNSA 1.0
Integrity	0 or 1	ESP512	-52	
Confidentiality	1	ECDH-ES + HKDF-512	-26	CNSA 1.0
Confidentiality	1	ECDH-SS + HKDF-512	-28	CNSA 1.0
Confidentiality	1	ECDH-ES + A256KW	-31	CNSA 1.0
Confidentiality	1	ECDH-SS + A256KW	-34	CNSA 1.0

Table 8: ECC Algorithms

When generating a BIB result from an ECC private key, 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 ECC private key, the top-layer headers SHALL include a corresponding public key identifier (see Section 3.5).

When generating a BCB result from an ECC 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 ECC public key, the recipient layer SHALL include a public key identifier (see Section 3.5). When a COSE_Encrypt is used with an ECC public key, the security source SHALL either generate an ephemeral ECC keypair or choose a unique HKDF "salt" for each security operation.

When a COSE_Encrypt is used with an ECC public key and a single recipient, the direct HKDF algorithms (code -25 through -28) are RECOMMENDED over the key wrapped algorithms (code -29 through -34) to reduce message size and the need for symmetric key generation. When processing a COSE_Encrypt with an ECC public key, a security verifier or acceptor SHALL process all KDF 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.

The choice of whether to use ECDH in static-static (SS) or ephemeral-static (EH) mode depends on what security properties are needed for the operation. ECDH-SS can reduce message size and allows key generation to happen outside of the source entity, but also requires the ECC public key to either be known by the recipient(s) and identified by or be fully transmitted by a header parameter (as discussed in Section 6.3.1 of [RFC9053]). ECDH-ES can provide forward secrecy by using the ephemeral key only for single messages, but also requires the source to generate a new key when needed.

3.2.4. RSA Algorithms

Implementations conforming to this specification SHALL support the RSA algorithms in Table 9 if they will operate with RSA key material.

> | The RSA-based algorithms are CNSA 1.0 conformant [CNSA1] only when used with a key modulus of 3072 bits or larger.

BPSec Block	COSE Layer	Name	Code	Conformance
Integrity	0 or 1	PS384	-38	CNSA 1.0
Integrity	0 or 1	PS512	-39	
Confidentiality	1	RSAES-OAEP w/ SHA-512	-42	CNSA 1.0

Table 9: RSA Algorithms

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].

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).

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 10. 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 10 uses the term "required" where the parameter needs to be understood by all message processors, "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) is conditional and chosen by the source.

Name	Label	Need
alg	1	Required for COSE [RFC9052]
crit	2	Required 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	Required for HKDF-using algorithms (direct and ECDH)
x5t-sender	-27	Conditional for ECDH-SS algorithms
x5chain-sender	-29	Conditional for ECDH-SS algorithms

Table 10: Interoperability Header Parameters

This profile of COSE does not use in-message KDF context information as defined in Section 5.2 of [RFC9053]. The context header parameters for PartyU (code -21 through -23) and PartyV (code -24 through -26) SHALL NOT be present in any COSE message within this security context. A side effect of this is that, to satisfy COSE requirements, the "salt" parameter SHALL always be present in a layer when an HKDF is used by the algorithm for that layer.

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. The profile here mandates specific data to be present in CA and end-entity certificates but does not mandate any specific key types or signing algorithms to be used. Such details are left to algorithm-specific profiles such as [RFC8603] for CNSA 1.0.

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. Security verifiers and acceptors SHALL use the Subject Alternative Name extension for identity information in end-entity certificates.

All BPSec end-entity certificates SHALL contain a Basic Constraints extension in accordance with Section 4.2.1.9 of [RFC5280] marked as critical.

All BPSec end-entity certificates SHALL contain a Subject Alternative Name extension in accordance with Section 4.2.1.1 of [RFC5280] marked as critical. 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 BPSec CA certificates 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]. All BPSec end-entity certificates SHOULD contain 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.

All BPSec CA certificates SHOULD contain an Extended Key Usage extension in accordance with Section 4.2.1.12 of [RFC5280]. When allowed by CA policy, a BPSec end-entity certificate SHALL contain an 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 Key Usage extension in accordance with Section 4.2.1.3 of [RFC5280] marked as critical. 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].

Specifically for the case of vulnerability to a cryptographically relevant quantum computer, algorithms for signing and key encapsulation have been identified in [CNSA2] but are not yet available as COSE code points allocated by published standards.

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. When public key certificates are used, all of the security considerations of [RFC5280] and any other narrowing PKIX profile 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 3) 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 11: 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 Special Keys". Table 12 shows the initial values for this registry.

The registration policy for this registry is Specification Required. Specifications of new entries need to define how they relate to AAD generation procedure of Section 2.5.1.

The value range is negative 16-bit integer. This value range is combined with the non-negative 64-bit integer block numbers for the AAD Scope key domain (Section 2.2.2).

Value	Name	Reference
-1	Target block	[This specification]
-2	Security block	[This specification]
-3 to -238	Unassigned	
-239 to -240	Reserved for Experimental Use	[This specification]
-241 to -256	Reserved for Private Use	[This specification]
-257 to -65536	Reserved	

Table 12: BPSec COSE AAD Scope Special Keys

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 13 shows the initial values for this registry.

The registration policy for this registry is Specification Required. Specifications of new entries need to define how they relate to AAD generation procedure of Section 2.5.1.

The value range is a bit position within an unsigned 64-bit integer.

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

Table 13: 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>.

[SDLS] Consultative Committee for Space Data Systems, "Space Data Link Security Protocol - Summary of Concept and Rationale", CCSDS 350.5-G-2, January 2024, <https://public.ccsds.org/Pubs/350x5g2.pdf>.

[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>.

[RFC8603] Jenkins, M. and L. Zieglar, "Commercial National Security Algorithm (CNSA) Suite Certificate and Certificate Revocation List (CRL) Profile", RFC 8603, DOI 10.17487/RFC8603, May 2019, <https://www.rfc-editor.org/info/rfc8603>.

[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
7372632f821b000000bd51281400001a000f42408501010000466568656c6c6fff
```

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 ECC 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 using key wrap or encapsulation (Appendix A.5, Appendix A.7, Appendix A.9) 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 384-bit symmetric key (same size of the hash output) identified by a "kid".

```
[
  {
    / kty / 1: 4, / symmetric /
    / kid / 2: 'ExampleMAC',
    / alg / 3: 6, / HMAC 384 384 /
    / ops / 4: [9, 10], / MAC create, MAC verify /
    / k / -1: h'3a5c74e32ab4558a99581ec3a816576812aabe895db04494cda2
5b711d7b5ed4077466e677860648412f1bf8c91d0624'
  }
]
```

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: 6 / HMAC 384 384 /
        }>>,
        { / unprotected /
          / kid / 4: 'ExampleMAC'
        },
        null, / payload detached /
        h'67d97d8c01c0656d0d2badac112990f05b3b285809cf2bd879d9498a7b
fdefa119ba9988a69874a3349b0451b5f2b029' / tag /
      ]>>
    ]
  ]
]
```

Figure 14: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 170 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850b03000058608101030182016 6
2f2f7372632f818205a200012001818182115845844 3a10106a1044a4578616d706c
654d4143f6583067d97d8c01c0656d0d2badac112990f05b3b285809cf2bd879d949
8a7bfdefa119ba9988a69874a3349b0451b5f2b029850101000046656 8656c6c6fff
```

A.2. ECC Keypair COSE_Sign1

This is an example of a signature with a recipient having a P-256 curve ECC 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: -51, / ESP384 /
    / ops / 4: [1, 2], / sign, verify /
    / crv / -1: 2, / P-384 /
    / x / -2: h'02dfc49747f5d3d219fe6185744729fa1672ef7d11cb57ca0320
c632be06ca3fdcc118e63140ba3ec57ea7b85d419568',
    / y / -3: h'4526e81bf0d9ea0924f05a3453ad75b92806671511544c993f6b
d908a7a4239d476cfdfd74d6c68836488ad1e60b0e7d',
    / d / -4: h'3494803544d85a84d802400b50f51eea23b72d7d850b53cbf300
6e5be2940d4a2c18d510a412efc7dc7875fbba22cca9'
  }
]
```

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: -51 / ESP384 /
           }>>,
           { / unprotected /
             / kid / 4: 'ExampleEC2'
           },
           null, / payload detached /
           h'd1eac4f875e19303a76dacaf9873f8dcfec26e30c573642d908ad1babd
   de44b35ff0776c765c1f2a44d2f69571263ed55f843686f69d00e078de686d720609
   212f435ca36283853803f5bc085192dc5649036611ff0c395b53b06e9d06382a
   2a' / signature /
         ]>>
       ]
     ]
   ]
```

Figure 17: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 219 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850b03000058918101030182016 6
2f2f7372632f818205a200012001818182125876844 4a1013832a1044a4578616d70
6c65454332f65860d1eac4f875e19303a76dacaf9873f8dcfec26e30c573642d908a
d1babdde44b35ff0776c765c1f2a44d2f69571263ed55f843686f69d00e078de686d
720609212f435ca36283853803f5bc085192dc5649036611ff0c395b53b06e9d0638
2a2a850101000046656 8656c6c6fff
```

A.3. RSA Keypair COSE_Sign1

This is an example of a signature with a recipient having a 3072-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: -38, / PS384 /
    / ops / 4: [1, 2], / sign, verify /
    / n / -1: h'f47b7c275eff5bf74e9e69d2b50e9a5dc0666147ca2541fce9a2
53d00840e2481378733797a9db641911937aff7cbc579f16484cea2cd3fbe2db3bbb
26b18e2b95b49b9255f378ee62f806227dd0ddeb48456358bfe1cadd861645e89208
c5a5b1da34af0e3f0ad1f82633fb6b21b7ef901bffebf1144ed0d4cfcb51e21b820c
fa6c829ca95cd08f3eb779b31569bb5e64c67f9976df92284ecea085a9a25882707a
9ba2af769017983e58b48afde78149453ebdcf1bcad76c30c253deaacc21a4a91bce
2b896776c1c5cfe0aaaae714c0b7e2063542ea81ec27af2746fd402e02e71040e41a
a23c79179e20a93c921fb26d13e11f802c2836506d0f0be507997aa07454f8b74a80
7c9342c8697d670fe6b1c2574a18013296cca897c4ddb96215fd30a4c4ee5db5b484
4c9d1e7bd6923ab0feaa30fe79d821fb63ed391b175837d53db1b207bc984b9a377c
7b4fb58738cca4254c3273440830cee84552304e8aaed0c0187b59339b28f213076a
f2c77aa46fe37bef31910ada5e44643ac7b1',
    / e / -2: h'010001',
    / d / -3: h'5a4dfac0d33317ff5b1879ab793db7455f2ce0e596fd5ff1e901
15a08087b18c413a15d4a007e41d8a71eaa81490a926bd4289d072e6b8102ba25073
1c02b238b95cc4f1d2d1ece45cb41c66a22bafcd0da48726daf2a20c159136e7d527
944f46d73ee084184682e4ca0a287d26e92535c532a4790e8bb6589f0bce2f6f693d
900db994f9fb92a95f01ed077cb667f938ab73f9cc27715bcc357fb4acb20630ad2c
40893b0a6a9be0ab50b7d4bfbd088e85919e61f325a33b0fc2e709c879c33ab7675a
395f45c8e5f08c1a7e1871f6bc12eef424566896eca111e42060ea64ea5ea11f693f
6923525829a0e394434eb0ec1a5586677bcab8a7b8d961b6134236f027d3dd2c0fee
03e46f534d782d73bf4b1b906dae037c4fc6982082b5c1946418780e3a2a14ba5717
ddb98df721e42da8b419a7febc06c3948cd0612e2b3d861077d5622af774d6af4f19
f1d38658174eded20954d45d4c1db848ea3c8805aaf338615ae9e40e3990352c0dfe
84563b67d2e72638806807e8b56edb41',
    / p / -4: h'fd804ae7f7499d69cd3d1299879e9645b90e829e92c88d759bf2
b3d4ed391b592407c2f1f26dd0f345bbe448b21a2248450652aebaa65a6815919dba
29d66ea2c1d862cab37391190c5e1fb162d7101412588d913fecd0e675785522661a
ecbbc50e2678e60485080b472e5f5720854a21eb456e1c26de4102d99cc2e08afa13
d2aa3e7f94d91e2fb43ab400c5e3061c69fcfcf78c7a81cdcc672d0f5630eb08ac90
86a8c957d4f8c9c71dd75a78efe1e1d764d34d7043f0aac9a0d0beaf8921',
    / q / -5: h'f6e46eff1e17493b92e379a6ae2594970d6ab3ff43bb84fda781
0b5feb89d94008fbf8de9ce8584aa95ad759261ce78fd421c26f5c1eaf482792e6fc
25eeac320a2195db28ab9071182ca4d4d041cfea6d754940248527352337e87e7b44
14d59b916808e977d97676cf97ee04b7f95aa3611f1fb98b7a341563ca42e9af810b
67ecd8989deb96ddde66004efdda610c77cd75b9337b5e1993c6d2b3212c64461d61
334193bad6ebb62ae67325e1157a394c8c7422a3d65ecd0212114d7e9c91',
    / dP / -6: h'ec2626217f38c17e3d26267c855d1389f2017566b940409f0dd
```

```
   e82edd8eb38f1ca61bc95dd0bb5f9d9bd55c4eebcefb0b93451b3d9c67c33b7dc05b
   dd5999f48d92175ae748b34e0cba7a7087d15f1317181b2a75b90856e1a823574acf
   f6a06e563f02cf1c1c6179f41f90df1c126c9cf5d373982da2673136f9adbe38733b
   d61a31c43876ad6f70383280a0c4e177442bbdcffd28a90ff20ea008ce7f2fc10018
   9451859300c02931d7d4c0f48d7d669a758928af209285a4128212d71a261',
       / dQ / -7: h'a0d29842f294f48d2bd7a56c9fcfb704d626856d67ef8467be6
   edebbf2afeea639b3f89ef9d29780bae483967caf235f9b2d0a7c83a331466d10d20
   9b9a3c8e3279a4d055f6eb23e19232b93bcbcc1f4d0ac2fb4ea9519bf115bdfc4540
   33b1711a91bfd822721ae7b222ab34ebb90602c409d878ad3821cdf3a0b8c9eb045f
   cea0b6be3ae2ac23170273d58371fc34bddd626332787dafa0a3adf10f430f8787bb
   6cf2e8e4e8ca52a1ab3d699fc0e83794395d228a6548398431b05ce570521',
       / qInv / -8: h'1f1b16da1f590fc9e6b174122d8c78554adee8627c31068ab
   ccf56f8a73c04fd677567d9fe246b1ff972c9b74e238bf4e04b9cef7e0ba76befea4
   3a0e114a0aff45b2cbeff649614281017c1f00be91ba2562453a0a5ee25f6518fcf0
   7dddf2da2c645bc337a51b8108dba1aab223893c7fcbabdb5b9c88b618e858eda994
   b7c04b1bffe2612f743e857707dccea4f7a93d711f818a8e6420890c2e73eb7f4fcc
   3c55c7d83b4d2bbd9bcea0c0668570e9ca7e92e5ca626754180da12a6b85a85'
     }
   ]
```

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: -38 / PS384 /
           }>>,
           { / unprotected /
             / kid / 4: 'ExampleRSA'
           },
           null, / payload detached /
           h'93573a924293bc39714337edebd6bc6ea5a3b8e70f765f1f0cbf1e9410
e890117d745858b6d63ec85414b7d9e37d5c83cd44b2ba58bb68b9ea5a9d988cb80c
47b0218f7f4f17427a7707f951703d90505613a5728ac53851cb230fa143fc622661
d0f2042a4d588ca7e2ba1009943258fe0181de637d8a3a7c0659906f6fbbda5c2941
81032831a07fe61aedefd862332417c08b4e8c7af6d5239425a6c93eb0d851ec9994
5ec80d3004eec0d9a129f4f2cc62f29548d429e4a6ea30fe07638e647b2c1b4c06a4
3af25b318f1d368957c76c1b1e3f60e123e7d51e45a3951a6ee9922343c02666cc51
852c95a3dee396f650f2b2af3f0e02d92a1793855397a55087e5b3eb25eb78e9493b
72ae8710069c62e78d3968983a2058c9d1242903bf97d2ffd803a989dcf8a80d514b
dc84365258fc30d4a29201e115c589664ed7ca27eddb0a5524902bcae2f1cd77ec24
44971e80596bb1f5b22ef4c53fe493b460d1bb449a95b645b10c1e65e593afea852b
9e6786ea7edd8b230e4a9c59e4468a' / signature /
         ]>>
       ]
     ]
   ]
```

Figure 20: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 510 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850b0300005901b3810103018201
662f2f7372632f818205a200012001818182125901978444a1013825a1044a457861
6d706c65525341f659018093573a924293bc39714337edebd6bc6ea5a3b8e70f765f
1f0cbf1e9410e890117d745858b6d63ec85414b7d9e37d5c83cd44b2ba58bb68b9ea
5a9d988cb80c47b0218f7f4f17427a7707f951703d90505613a5728ac53851cb230f
a143fc622661d0f2042a4d588ca7e2ba1009943258fe0181de637d8a3a7c0659906f
6fbbda5c294181032831a07fe61aedefd862332417c08b4e8c7af6d5239425a6c93e
b0d851ec99945ec80d3004eec0d9a129f4f2cc62f29548d429e4a6ea30fe07638e64
7b2c1b4c06a43af25b318f1d368957c76c1b1e3f60e123e7d51e45a3951a6ee99223
43c02666cc51852c95a3dee396f650f2b2af3f0e02d92a1793855397a55087e5b3eb
25eb78e9493b72ae8710069c62e78d3968983a2058c9d1242903bf97d2ffd803a989
dcf8a80d514bdc84365258fc30d4a29201e115c589664ed7ca27eddb0a5524902bca
e2f1cd77ec2444971e80596bb1f5b22ef4c53fe493b460d1bb449a95b645b10c1e65
e593afea852b9e6786ea7edd8b230e4a9c59e4468a850101000046656865 6c6c6fff
```

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 Key COSE_Encrypt with Key Wrap

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'
  },
  { / wrapped CEK /
    / kty / 1: 4, / symmetric /
    / alg / 3: 3, / A256GCM /
    / 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 /
               <<>>, / 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. Symmetric Key COSE_Encrypt with HKDF

This is an example of an encryption with a derived CEK and an explicit key-derivation key (KDK) identified by a "kid". The keys used are shown in Figure 28, where the second key is the CEK derived from the KDK via a salt value in the recipient header.

```
[
  {
    / kty / 1: 4, / symmetric /
    / kid / 2: 'ExampleKDK',
    / alg / 3: -11, / direct+HKDF-SHA-512 /
    / ops / 4: [7], / derive key /
    / k / -1: h'0e8a982b921d1086241798032fedc1f883eab72e4e43bb2d11cf
ae38ad7a972e'
  },
  { / derived CEK /
    / kty / 1: 4, / symmetric /
    / alg / 3: 3, / A256GCM /
    / k / -1: h'b0f3604f54d3f15e272357d1df75b50ede1475b0cdb17fef0fd1
46b9bbcda3a3'
  }
]
```

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 31. This ciphertext is different than the common one in Figure 11 because of the different derived CEK in Figure 28.

```
   [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: -11 / direct+HKDF-SHA-512 /
               }>>,
               { / unprotected /
                 / kid / 4: 'ExampleKDK',
                 / salt / -20: h'2fa8c8352aea17faf7407271a5e90eb8'
               },
               h'' / empty /
             ]
           ]
         ]>>
       ]
     ]
   ]
```

Figure 30: Abstract Security Block CBOR diagnostic

```
   [
     1, / type code: payload /
     1, / block num /
     0, / flags /
     0, / CRC type /
     h'7443d510ee5ce1280ac6639b2256c98015ebfa8a8eb4' / ciphertext /
   ]
```

Figure 31: Encrypted Target block CBOR diagnostic

The final bundle is encoded as the following 177 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850c0300005857810103018201662
2f2f7372632f818205a200012001818181860583b8443a10103a1054c6f3093eba5
d85143c3dc484af6818343a1012aa2044a4578616d706c654b454b33502fa8c8352a
ea17faf7407271a5e90eb840850101000056744 3d510ee5ce1280ac6639b2256c980
15ebfa8a8eb4ff
```

A.7. ECC Keypair COSE_Encrypt with Key Wrap

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 32.

```
[
  { / sender ephemeral private key /
    / kty / 1: 2, / EC2 /
    / crv / -1: 2, / P-384 /
    / x / -2: h'2f88f095c45c96e377e18d717a5e6007ce8f6076ae82009d1637
5e1b9abaa9497a4bde513be6c9b0e7dae96033968c45',
    / y / -3: h'fd27656fbb97f789d667f40d73b65ab362b22dd23bf492bee72b
f3409f68dddf208040a5fcbcbee74545741e2866cb2d',
    / d / -4: h'c4fff15193b8bceff5e221cc37b919fa8d33581a37c08d3e8520
a658b4040a443f8fb3b54fb4ce882510e76017b66261'
  },
  { / recipient private key /
    / kty / 1: 2, / EC2 /
    / kid / 2: 'ExampleEC2',
    / alg / 3: -31, / ECDH-ES + A256KW /
    / ops / 4: [7], / derive key /
    / crv / -1: 2, / P-384 /
    / x / -2: h'0057ea0e6fdc50ddc1111bd810eae7c0ba24645d44d4712db0c8
354c234b2970b4ac27e78f38250069d128f98e51ceb1',
    / y / -3: h'4b72c50b27267637c40adcd78bd025e4b654a645d2ba7ba9894c
c73b2431d4cdc040d66e8eb2dad731f7dca57108545c',
    / d / -4: h'7931af7cc3010ae457bcb8be100acdafab8492de633b20384c3e
4de5e5e94899d9d9de25c04d6205ae6bb9385ce16ff7'
  },
  { / wrapped CEK /
    / kty / 1: 4, / symmetric /
    / alg / 3: 3, / A256GCM /
    / k / -1: h'13bf9cead057c0aca2c9e52471ca4b19ddfaf4c0784e3f3e8e39
99dbae4ce45c'
  }
]
```

Figure 32: Example Keys

The external_aad is the encoded data from Figure 10.

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

Figure 33: Enc_structure CBOR diagnostic

The ASB item for this encryption operation is shown in Figure 34 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: 2,
                   -2: h'2f88f095c45c96e377e18d717a5e6007ce8f6076ae8200
   9d16375e1b9abaa9497a4bde513be6c9b0e7dae96033968c45',
                   -3: h'fd27656fbb97f789d667f40d73b65ab362b22dd23bf492
   bee72bf3409f68dddf208040a5fcbcbee74545741e2866cb2d'
                 }
               },
               h'0eaff015e61418d8910ba25ed9733450558b6a20ab410f3c925b01
   ac8d3aefcc12433f9563da401d' / key-wrapped /
             ]
           ]
         ]>>
       ]
     ]
   ]
```

Figure 34: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 309 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850c03000058db81010301820166
2f2f7372632f818205a200012001818182186058bf8443a10103a1054c6f3093eba5
d85143c3dc484af6818344a101381ea2044a4578616d706c6545433220a401022002
2158302f88f095c45c96e377e18d717a5e6007ce8f6076ae82009d16375e1b9abaa9
497a4bde513be6c9b0e7dae96033968c45225830fd27656fbb97f789d667f40d73b6
5ab362b22dd23bf492bee72bf3409f68dddf208040a5fcbcbee74545741e2866cb2d
58280eaff015e61418d8910ba25ed9733450558b6a20ab410f3c925b01ac8d3aefcc
12433f9563da401d8501010000561fd25f64a2ee886e97ecfde7667371214f5add54
a089ff
```

A.8. ECC Keypair COSE_Encrypt with HKDF

This is an example of an encryption with an P-256 curve static sender keypair and a static recipient keypair each identified by a "kid". The keys used are shown in Figure 35, where the third key is the CEK derived from the ECDH secret via a salt value in the recipient header.

```
[
  {
    / kty / 1: 2, / EC2 /
    / kid / 2: 'SenderEC2',
    / alg / 3: -28, / ECDH-SS + HKDF-512 /
    / ops / 4: [7], / derive key /
    / crv / -1: 2, / P-384 /
    / x / -2: h'2f88f095c45c96e377e18d717a5e6007ce8f6076ae82009d1637
5e1b9abaa9497a4bde513be6c9b0e7dae96033968c45',
    / y / -3: h'fd27656fbb97f789d667f40d73b65ab362b22dd23bf492bee72b
f3409f68dddf208040a5fcbcbee74545741e2866cb2d',
    / d / -4: h'c4fff15193b8bceff5e221cc37b919fa8d33581a37c08d3e8520
a658b4040a443f8fb3b54fb4ce882510e76017b66261'
  },
  { / recipient private key /
    / kty / 1: 2, / EC2 /
    / kid / 2: 'ExampleEC2',
    / alg / 3: -28, / ECDH-SS + HKDF-512 /
    / ops / 4: [7], / derive key /
    / crv / -1: 2, / P-384 /
    / x / -2: h'0057ea0e6fdc50ddc1111bd810eae7c0ba24645d44d4712db0c8
354c234b2970b4ac27e78f38250069d128f98e51ceb1',
    / y / -3: h'4b72c50b27267637c40adcd78bd025e4b654a645d2ba7ba9894c
c73b2431d4cdc040d66e8eb2dad731f7dca57108545c',
    / d / -4: h'7931af7cc3010ae457bcb8be100acdafab8492de633b20384c3e
4de5e5e94899d9d9de25c04d6205ae6bb9385ce16ff7'
  },
  { / derived CEK /
    / kty / 1: 4, / symmetric /
    / alg / 3: 3, / A256GCM /
    / k / -1: h'67bb109aaee51e9616b512d5750139444ca26e6c0eeaa87f3917
de41dd9ad9f6'
  }
]
```

Figure 35: Example Keys

The external_aad is the encoded data from Figure 10.

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

Figure 36: Enc_structure CBOR diagnostic

The ASB item for this encryption operation is shown in Figure 37 and corresponds with the updated target block (containing the ciphertext) of Figure 38. This ciphertext is different than the common one in Figure 11 because of the different derived CEK in Figure 35.

```
[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: -28 / ECDH-SS + HKDF-512 /
            }>>,
            { / unprotected /
              / kid / 4: 'ExampleEC2',
              / sender kid / -3: 'SenderEC2',
              / salt / -20: h'2fa8c8352aea17faf7407271a5e90eb8'
            },
            h'' / empty /
          ]
        ]
      ]>>
    ]
  ]
]
```

Figure 37: Abstract Security Block CBOR diagnostic

```
[
  1, / type code: payload /
  1, / block num /
  0, / flags /
  0, / CRC type /
  h'925c33206eb8223957ae40693c02d1f6a0cdb71574aa' / ciphertext /
]
```

Figure 38: Encrypted Target block CBOR diagnostic

The final bundle is encoded as the following 189 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850c0300005863810103018201662
2f2f7372632f818205a200012001818182186058478443a10103a1054c6f3093eba5
d85143c3dc484af6818344a101381ba3044a4578616d706c65454332224953656e64
657245433233502fa8c8352aea17faf7407271a5e90eb840850101000056925c3320
6eb8223957ae40693c02d1f6a0cdb71574aaff
```

A.9. 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: -42, / RSAES-OAEP w SHA-512 /
    / ops / 4: [1, 2], / sign, verify /
    / n / -1: h'c257bd2257000620d6c12e7fb988b891c0fa0f8eab597e2c56f0
ae49da3f24eb9b99f9cc147820bd7f3feffc41cd63aa7a805a454c73d60cee478ac8
e6be050943a134565c5b84aa619ba674f901314e2007bab3740c3b581720bc89d50c
81726722d5c6bbb9966285dd66d4524561aadd0d6e8d7d970530a6c30af202e55e4e
44505ce08ebe5217d2825edfa76397226ef58517abcc2e73873386ef9a0d14306466
f2ed2c61ce6eaca12a026b62db054379e9f6575e802355f53ce3efd44d58dd9f2ba7
385130815eed0e4649547fc38ff469a4f098d22214a6adf72c3f6a3b3ee3545835ea
a9a9ab6467b17a62acb3179f31dc534539ffd21981fe5e5ea088ff4f1cdf051d8d97
704101e81d8030e2e4863c1571452f94a9f47ec7339536058c287c376b0a6b5c6226
fdefd716b4438a9f987d50de25a73537d42a54d9d042f8d623f493d1fe0fdea8cd75
0381796cc9af7fdbac6eb7c8b4aa2e3f227d0fbbfbece7fd707e0f739b23a63b5a51
18f553e834facda493e276ec9663ba65bf2b',
    / e / -2: h'010001',
```

```
      / d / -3: h'132665ced99cce7dc8e643488c5b37b8b662784afa2bce33874e
   486ba2e9004b2e762c8d0562aaf33bf3ec6d6d2729d02ad45ffb73ce4c442797541f
   cf7634f52b3d5a1f67bd658eb6773473fd4a0bf638a61a546ee07a5932a4e3ff299d
   05afba104ed964e12390f4c392cb8edf2301c7ae22fbd294218ba03b183bd8638aa3
   3d61b52d3428f684e8c0e0f6ba934fd95c440432876d63670de65ac05c6be2813c3b
   8014dfd53416bd7054bd3aa0af42a45a01df12253a8cd62908057be4987462758517
   12623580de4cea4cefb2fbd154874bf13894d2a09ea9f18a9d6fcf66e8fefbf4029c
   e635ac1b60b444bab134ec67e397172a1dc79018ccc5713edaa21917cb34486a2a81
   c9e45ee21eb9bf465e7f42b4dfe17f17505b07d52bcf5233f215d96479201858e238
   b2672c2de6c589107f2877479a39728360a718400b3de94759f4e50049e279677481
   dd9151f82be9fef8289e297123e0ef08479010c4cc98096788857009fa41f8670556
   03491dea29e74d30925b42cceb004e7a371d',
      / p / -4: h'e49562b275ab259750b010ade9720f0c1e07c42e73d969b9c28d
   f2bb6a72a162c24101a9a97091a935ce5202015e9540971734ceb0f4014fecbaf7f0
   c4a6eeb6d82b46a69b4e5ed9a1099035e6dbfa3686985f8ae9c6a2f1afd2c1a5b60d
   16f4b1d22741bcfb12c103c11a3a68700c93e010906192289774f86400bbd513fad6
   42b32fb7b9ddb05d5c48ce9bbcae48239ce41f7630f6ec6ccf9dbeffc84edc2944f7
   49b16f445b829a7bc5dce644a377fa88035e0756aeca77c71dcb8f87514f',
      / q / -5: h'd9a6fee7a8e74201b5afe5188b0498fe4f902b7a4fd18fe64c1d
   6344188bb4ac26ccf3ff66f244ab2bb2d87fc94ac9e3e13482799f12585df2b6d556
   233c818853071912bc56c2b4c81ef9674e552af7bf8907f9ff9d318dcf7bc04eb086
   4a6c618468e3005721c1b9de436f81e9f9ae5d54228eba78af72760997e91d6f9481
   a43a5557fce42acf08868b460cfca2f3cdee7f47205f24343bceebb011e4aa80f94c
   c3a65f04dd5810e783d509f2346488338ec9012a046ad92ea9a10589e565',
      / dP / -6: h'a8b520fd3a1fb144f7069ba8e02d90b18ed08899086424c637b
   3f0bd2699a8476dbbf0f039e09d8157f7094bf59acb69ba9a241d9138e66709000dd
   3243158ea96ad8a1d996ec44eb7ae89435f3a687829eaf8495cb580ba04dcf693c9c
   3eb777a6ef30e6fde973ee1f879d53613cd14af414a6ed9232075f2864c8c557dc39
   ab3ebf08217aa696ade9bd5f1d7d681e3d6fdffc289ed151e5235c92c9bb8a881c52
   706baf0b6711bf9ccf4824f69c584dde1d92a631c3531b629bdf1e9e323bd',
      / dQ / -7: h'a0fa7a9e2cb69e835535fb63e3ae4ada0d4ebc59829fa4a6d8b
   503ae61d932900142a554c977768283978bb937d030f272a6bbb9e88551066b75fee
   3eebbd9b25276757cfdffcd9298511075efe1de1dcf74328a1d1cce81ec6bc318704
   762d4366c108794c0dd1ec3b2387e48c01d0371d3c09b801fb2e41d998ad9c803b6f
   b0bd4793ad2b88f51012541ed55bda5685d6f8083c2d59b996682ec9f151ce35ef10
   46dd0a786998f81313ab85edadd155e07841bf6d874dbf23629100760ae61',
      / qInv / -8: h'598da6c558bf08c201b845b2dab3ff00a2ee74ce064d86f18
   af2f8b721205224526b7d8b9c42f6bc7f34a8c8623dcfc28ff800e28f23cd3018148
   57a728b282a121ee6d47d031eef5c14d84d6aadfd2bdf3ef9d10dce7dda11ba466f1
   25d67772b945b79baa5092f86f98dcfd1d8fc946fd24851bc3e49033c29d6509a73d
   64326d3981b165be7bb2fa15d2696200c786fe1098449ded9207af0391caabf617da
   3fd8c777e1ad755bd24855dc6d84933987543f12fba160c3c71de8bff439468'
     },
     { / encapsulated CEK /
       / kty / 1: 4, / symmetric /
       / alg / 3: 3, / A256GCM /
       / k / -1: h'13bf9cead057c0aca2c9e52471ca4b19ddfaf4c0784e3f3e8e39
   99dbae4ce45c'
```

```
    }
  ]
```

Figure 39: Example Keys

The external_aad is the encoded data from Figure 10.

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

Figure 40: Enc_structure CBOR diagnostic

The ASB item for this encryption operation is shown in Figure 41 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 /
               <<>>, / protected /
               { / unprotected /
                 / alg / 1: -42, / RSAES-OAEP w SHA-512 /
                 / kid / 4: 'ExampleRSA'
               },
               h'407e01f5fc110dcf186fb9d7b627a45a2cfb230724b55a7b17aba0
   e184688c7913278b505d23d9a0087fc02bc3422b8e12d106741ce1e075fa0346352d
   f5c4a4f7e81c4e2c08003fb1df95f8e1b84b92e06d23b8c23b50e3507635df98841c
   c5ec190b1095919515886476ba674a0fccad15e367a2b7515be59926abf5be8082ff
   68679446c6f95dc05366014d437740e1878396a9e25dfb11000e5f07a45f9e3bebd7
   b4a2f7d28313de67d41b9ac11873285140244aa0df61cdfcc02e4b06e1493814d57d
   4748c035535caccfdd897066c21efc7f79abfd070297b991fb049a86114138c68f46
   47fe9052d8378cff99a7bdbe340041cc272e4d612b62ca4f1892f022164095ff63d6
   32d55b66db7e025ec28374083d933f53d8d9750bfba377c8e0afdc3636960e19703e
   1b65c1c7684b2543c252f086f37792fd65935f6dc942acfea7cad77ae2777ff3f700
   f37dfeb65f3968a4fc7544d2d346e498f8f53737877583387846bae1baaef0aab6dd
   5725e617d0d994cc0c8a1a3a6b659f768a' / key-encapsulation /
             ]
           ]
         ]>>
       ]
     ]
   ]
```

Figure 41: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 547 octets in base-16:

```
9f880700008201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f4240850c0300005901c8810103018201
662f2f7372632f818205a200012001818218605901ab8443a10103a1054c6f3093
eba5d85143c3dc484af6818340a2013829044a4578616d706c65525341590180407e
01f5fc110dcf186fb9d7b627a45a2cfb230724b55a7b17aba0e184688c7913278b50
5d23d9a0087fc02bc3422b8e12d106741ce1e075fa0346352df5c4a4f7e81c4e2c08
003fb1df95f8e1b84b92e06d23b8c23b50e3507635df98841cc5ec190b1095919515
886476ba674a0fccad15e367a2b7515be59926abf5be8082ff68679446c6f95dc053
66014d437740e1878396a9e25dfb11000e5f07a45f9e3bebd7b4a2f7d28313de67d4
1b9ac11873285140244aa0df61cdfcc02e4b06e1493814d57d4748c035535caccfdd
897066c21efc7f79abfd070297b991fb049a86114138c68f4647fe9052d8378cff99
a7bdbe340041cc272e4d612b62ca4f1892f022164095ff63d632d55b66db7e025ec2
8374083d933f53d8d9750bfba377c8e0afdc3636960e19703e1b65c1c7684b2543c2
52f086f37792fd65935f6dc942acfea7cad77ae2777ff3f700f37dfeb65f3968a4fc
7544d2d346e498f8f53737877583387846bae1baaef0aab6dd5725e617d0d994cc0c
8a1a3a6b659f768a8501010000561fd25f64a2ee886e97ecfde7667371214f5add54
a089ff
```

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-SHA384
   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: (384 bit)
           pub:
               04:cc:7b:ba:7b:04:77:e0:f7:97:30:40:a1:83:fd:
               0c:8b:44:9f:6f:e2:bd:ab:ec:df:9c:7a:72:e2:2c:
               b3:55:6a:49:64:89:ca:75:f8:09:f1:1f:73:7e:08:
               00:71:c0:e6:1c:06:36:15:68:c2:24:be:ab:29:17:
               54:fd:40:c8:75:b8:be:3f:f7:46:0b:50:d4:28:1b:
               ec:95:d5:34:b4:4a:f4:97:71:5a:09:52:11:e3:59:
               28:b2:fb:f4:55:c7:6a
           ASN1 OID: secp384r1
           NIST CURVE: P-384
   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:
           1B:77:33:BE:83:75:66:6A:75:86:22:F2:AB:0A:17:60:3F:42:56:03
       X509v3 Authority Key Identifier:
           1B:77:33:BE:83:75:66:6A:75:86:22:F2:AB:0A:17:60:3F:42:56:03
```

Figure 42: CA Certificate Content

```
-----BEGIN CERTIFICATE-----
MIIB8DCCAXagAwIBAgIKFRX/p0CkvXP1ujAKBggqhkjOPQQDAzAgMR4wHAYDVQQD
DBVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwHhcNMjUxMDA2MDAwMDAwWhcNMjUxMDE2
MDAwMDAwWjAgMR4wHAYDVQQDDBVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwdjAQBgcq
hkjOPQIBBgUrgQQAIgNiAATMe7p7BHfg95cwQKGD/QyLRJ9v4r2r7N+cenLiLLNV
aklkicp1+AnxH3N+CABxwOYcBjYVaMIkvqspF1T9QMh1uL4/90YLUNQoG+yV1TS0
SvSXcVoJUhHjWSiy+/RVx2qjezB5MBIGA1UdEwEB/wQIMAYBAf8CAQAwDgYDVR0P
AQH/BAQDAgEGMBMGA1UdJQQMMAoGCCsGAQUFBwMjMB0GA1UdDgQWBBQbdzO+g3Vm
anWGIvKrChdgP0JWAzAfBgNVHSMEGDAWgBQbdzO+g3VmanWGIvKrChdgP0JWAzAK
BggqhkjOPQQDAwNoADBlAjBQLyBu8JDNdPcOkHpJZuH9BIbshDBEn3H+SNBubiS9
sRgqWp+gphgvVUBlo+na0TACMQCv0zQ7tVQHG7n8i3fw6hLNrk4UrwfXX91tcp3M
a9Z6MI8EU1mRAmqkM63oRHeNGS0=
-----END CERTIFICATE-----
```

Figure 43: CA Certificate PEM

```
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDBj90cnyONTJ3DqsSBdr4Df0zZ951wOLbQgqDPC8zw0wcrrQ5CT6+Ov
sA2i87696dWgBwYFK4EEACKhZANiAATMe7p7BHfg95cwQKGD/QyLRJ9v4r2r7N+c
enLiLLNVaklkicp1+AnxH3N+CABxwOYcBjYVaMIkvqspF1T9QMh1uL4/90YLUNQo
G+yV1TS0SvSXcVoJUhHjWSiy+/RVx2o=
-----END EC PRIVATE KEY-----
```

Figure 44: 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-SHA384
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: (384 bit)
        pub:
            04:02:df:c4:97:47:f5:d3:d2:19:fe:61:85:74:47:
            29:fa:16:72:ef:7d:11:cb:57:ca:03:20:c6:32:be:
            06:ca:3f:dc:c1:18:e6:31:40:ba:3e:c5:7e:a7:b8:
            5d:41:95:68:45:26:e8:1b:f0:d9:ea:09:24:f0:5a:
            34:53:ad:75:b9:28:06:67:15:11:54:4c:99:3f:6b:
            d9:08:a7:a4:23:9d:47:6c:fd:fd:74:d6:c6:88:36:
            48:8a:d1:e6:0b:0e:7d
        ASN1 OID: secp384r1
        NIST CURVE: P-384
X509v3 extensions:
    X509v3 Basic Constraints: critical
        CA:FALSE
    X509v3 Subject Alternative Name: critical
        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 Authority Key Identifier:
        1B:77:33:BE:83:75:66:6A:75:86:22:F2:AB:0A:17:60:3F:42:56:03
```

Figure 45: Signing Certificate Content

```
-----BEGIN CERTIFICATE-----
MIIB4TCCAWegAwIBAgIKb/6J3Ldu03LqejAKBggqhkjOPQQDAzAgMR4wHAYDVQQD
DBVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwHhcNMjUxMDA2MDAwMDAwWhcNMjUxMDE2
MDAwMDAwWjAOMQwwCgYDVQQDDANzcmMwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAAQC
38SXR/XT0hn+YYV0Ryn6FnLvfRHLV8oDIMYyvgbKP9zBGOYxQLo+xX6nuF1BlWhF
Jugb8NnqCSTwWjRTrXW5KAZnFRFUTJk/a9kIp6QjnUds/f101saINkiK0eYLDn2j
fjB8MAwGA1UdEwEB/wQCMAAwJgYDVR0RAQH/BBwwGqAYBggrBgEFBQcIC6AMFgpk
dG46Ly9zcmMvMA4GA1UdDwEB/wQEAwIHgDATBgNVHSUEDDAKBggrBgEFBQcDIzAf
BgNVHSMEGDAWgBQbdzO+g3VmanWGIvKrChdgP0JWAzAKBggqhkjOPQQDAwNoADBl
AjBHljyxGGWxBmV5pz6Mgkn2k8MH9Am0+4ZGzRcEvMORA9R6371sJ0OYpuy1pPrd
rwcCMQDrxYHocIePcAKYQnAAaNbn4pm/GaiTFgoQJWQn1tTMy3CyeocQMB0if57Y
w6Xw0+Y=
-----END CERTIFICATE-----
```

Figure 46: Signing Certificate PEM

B.3. Encryption Recipient End-Entity Certificate

This end-entity certificate corresponds with the private key used for decrypting Appendix A.7. 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-SHA384
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: (384 bit)
        pub:
            04:00:57:ea:0e:6f:dc:50:dd:c1:11:1b:d8:10:ea:
            e7:c0:ba:24:64:5d:44:d4:71:2d:b0:c8:35:4c:23:
            4b:29:70:b4:ac:27:e7:8f:38:25:00:69:d1:28:f9:
            8e:51:ce:b1:4b:72:c5:0b:27:26:76:37:c4:0a:dc:
            d7:8b:d0:25:e4:b6:54:a6:45:d2:ba:7b:a9:89:4c:
            c7:3b:24:31:d4:cd:c0:40:d6:6e:8e:b2:da:d7:31:
            f7:dc:a5:71:08:54:5c
        ASN1 OID: secp384r1
        NIST CURVE: P-384
X509v3 extensions:
    X509v3 Basic Constraints: critical
        CA:FALSE
    X509v3 Subject Alternative Name: critical
        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 Authority Key Identifier:
        1B:77:33:BE:83:75:66:6A:75:86:22:F2:AB:0A:17:60:3F:42:56:03
```

Figure 47: Key-Agreement Certificate Content

```
-----BEGIN CERTIFICATE-----
MIIB4DCCAWegAwIBAgIKPyQLzab3/Dwp3jAKBggqhkjOPQQDAzAgMR4wHAYDVQQD
DBVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwHhcNMjUxMDA2MDAwMDAwWhcNMjUxMDE2
MDAwMDAwWjAOMQwwCgYDVQQDDANkc3QwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAAQA
V+oOb9xQ3cERG9gQ6ufAuiRkXUTUcS2wyDVMI0spcLSsJ+ePOCUAadEo+Y5RzrFL
csULJyZ2N8QK3NeL0CXktlSmRdK6e6mJTMc7JDHUzcBA1m6OstrXMffcpXEIVFyj
fjB8MAwGA1UdEwEB/wQCMAAwJgYDVR0RAQH/BBwwGqAYBggrBgEFBQcIC6AMFgpk
dG46Ly9kc3QvMA4GA1UdDwEB/wQEAwIDCDATBgNVHSUEDDAKBggrBgEFBQcDIzAf
BgNVHSMEGDAWgBQbdzO+g3VmanWGIvKrChdgP0JWAzAKBggqhkjOPQQDAwNnADBk
AjArcmaF95pLvgjxXBYa7mtDhEEgnYVsZytcWFu74yLx/7u/mUEsK0AgOrV+uTTo
pqoCMAINw25QZUv9t8r+7lEmAo1em5730riu0Axq1yv0jF0LebLSYP6/fWe0cCwt
/zk1CA==
-----END CERTIFICATE-----
```

Figure 48: Key-Agreement Certificate PEM

Appendix C. CDDL Definitions for BPSec

The normative definitions of BPSec [RFC9172] do not include corresponding CDDL extending the rules defined for BP. The following CDDL provides those definitions as an update to that specification. These definitions include a new socket $ext-data-asb for all possible ASB contents and a generic rule bpsec-context-use which allows a security context to define a single rule for the ASB socket to include all of their parameter and result types together.

```
; Generic structure of block-type-specific data for BIB and BCB
ext-data-asb = $ext-data-asb .within ext-data-asb-structure
ext-data-asb-structure = [
  targets: [+ target-block-num],
  context-id: int,
  asb-flags,
  security-source: eid,
  ; params present if sec-params-present is set in #asb-flags
  ? parameters: asb-id-value-list,
  ; One result list per item in #targets
  target-results: [+ asb-id-value-list]
]
target-block-num = uint
asb-flags = uint .bits asb-flag-bits
asb-flag-bits = &(
  sec-params-present: 0
)

; Alternatives can be added to the sockets for each context ID
asb-id-value-list = [* asb-id-value-pair]
; Interpretation of the pair depends on the context-id and whether
; it is a parameter or a result.
```

```
asb-id-value-pair = [
  id: uint,
  value: any
]

; Provide BPv7 extension block types, they both really embed
; "ext-data-asb" as a cbor sequence.
; Block Integrity Block (BIB)
$extension-block /= extension-block-use<
  11,
  bstr .cborseq ext-data-asb
>
; Block Confidentiality Block (BCB)
$extension-block /= extension-block-use<
  12,
  bstr .cborseq ext-data-asb
>

; Specialization of $ext-data-asb for a security context.
; The ParamPair and ResultPair should be sockets for specializing
; those structures for the individual security context.
bpsec-context-use<ContextId, ParamPair, ResultPair> = [
  targets: [
    + target-block-num
  ],
  context-id: ContextId,
  asb-flags,
  ? security-source: eid,
  ? parameters: [
    + ParamPair .within asb-id-value-pair
  ],
  target-results: [
    + [
      + ResultPair .within asb-id-value-pair
    ]
  ]
]
```

Acknowledgments

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

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