Delay-Tolerant Networking
Internet-Draft
Updates: 9172 (if approved)
Intended status: Standards Track
Expires: 9 July 2026

B. Sipos
JHU/APL

5 January 2026

Bundle Protocol Security (BPSec) COSE Context
draft-ietf-dtn-bpsec-cose-14

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.

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, block-control-flags, the socket $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 results 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 parameters (defined in Section 2.2) to carry information applicable to all security operations and results (defined in Section 2.3) containing a specific COSE message for each security operation.

```
; 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 kind of data. COSE distinguishes between its internal portion of AAD, derived from COSE message content, and _external AAD_ provided by the embedding application, which in this case is the BPSec security context.

The sources for external AAD within this 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 the Default Security Contexts [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 block-type-specific data (BTSD) appears in the same bundle that the security source intended.

Other Canonical Block BTSD:
: Including the BTSD of an other, non-target block as part of AAD ensures that that other block’s BTSD does not change after the security operation is added. This can guarantee that not only has the security target BTSD not changed but the additional blocks’ BTSD have not changed.

Other 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 operation 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 (as described below), which binds the security result to that specific target block.

Target Block Metadata:
One special case of including block metadata as AAD is for the target block itself, which ensures that the target BTSD is bound to its specific containing block. This case uses AAD Scope key -1 and the value flag for metadata to indicate that the block metadata is taken from the target of the security operation.

Containing Security Block Metadata:
Another special case of including block metadata is for the security block containing the security operation itself, which ensures that the security operation is bound to its specific containing block. This case uses AAD scope key -2 and the value flag for metadata to indicate that the block metadata is taken from the containing security block.

Containing Security Block BTSD:
The BTSD content of the security block itself (as defined in Section 3.6 of [RFC9172]) is also partially covered by AAD as explained below.

* The Security Targets field can be included indirectly by using AAD scope key -1 to ensure the AAD includes each 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 always included as external AAD, so is protected from modification.

* The Security Context Flags and Security Context Parameters are not all included directly, but the modification of parameters will cause processing of security operations to fail. The Additional Protected parameter is the portion of this data which is included in the external AAD.

* The Security Results are also not included directly, but these are the COSE messages themselves which are designed to be handled as plaintext. There are portions of each COSE message (result) which is included in the internal AAD (via MAC_structure, Sig_structure, or Enc_structure) as defined by COSE [RFC9052].

Because of these options, 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 its single target block independently of the block metadata or bundle primary block associated with the target at the time the BIB is added. Likewise, it is possible for a COSE context confidentiality operation to be bound to every other block of a bundle at the time the BCB is added or bound to no context outside the BTSD of the target block.

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 instance of each parameter ID. Security verifiers and acceptors SHALL treat a security block with multiple instances of any parameter ID as invalid. There is no well-defined behavior for a security acceptor to handle multiple Additional Protected or AAD Scope parameters.

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 (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.

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 in the security block (across all security targets). 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

Each COSE context result contains a COSE message, but 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.

Additionally, this context restricts each security operation (embodied as a security target with a result array) to be associated with only a single COSE message (_i.e._, a single result for each target). Within the security context defined in this section, each

abstract security block SHALL contain exactly one result per security target. Security verifiers and acceptors SHALL treat other combinations of results-per-target as invalid. Security policies can be tailored for this constraint, which has a side effect that each security operation also has only a single set of COSE layer 0 header parameters (_e.g._, COSE algorithm).

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 verifier or 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.

It is expected, but not required, that keys referenced and used by COSE messages in this context will themselves be managed as COSE Key objects as defined in Section 7 of [RFC9052]. Using native COSE Key objects simplifies the work of an implementation to align with the key and credential identifiers contained in COSE header parameters.

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) generated as defined in Section 2.5.1 and the detached payload being the BTSD content from the target block as defined in Section 2.5.2.

2.5.1. Generating External AAD

The COSE external AAD content defined in this section are used for both integrity and confidentiality messages. The encoding of this content 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 scope (Section 2.2.2) 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.

This external AAD SHALL be encoded in accordance with the core deterministic encoding requirements of Section 4.2.1 of [RFC8949]. The external AAD content 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. This is a CBOR array of length 2 in accordance with Section 4.2.5.1 of [RFC9171].

2. The second part SHALL be the encoded AAD Scope value itself. This is a CBOR map in accordance with Section 2.2.2. 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 is the full primary block, including its definite-length array head. This part will be identical to the encoded primary block from the containing bundle if that primary block conforms to encoding requirements of Section 4.3.1 of [RFC9171].

        * 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 is just the three encoded CBOR unsigned 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 SHALL be the re-encoded BTSD of the block identified by the block number in the map key. This is a definite-length CBOR byte string. This part will be identical to the encoded BTSD item from the target block itself if that target block conforms to encoding requirements of Section 4.3.2 of [RFC9171].

4. The last part SHALL be the encoded form of the Additional Protected parameter (Section 2.2.1). This is a definite-length CBOR byte string. 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 the CBOR head in an encoded form is ignored for the purposes of security processing; only the BTSD content bytes are significant. This also means that if the target BTSD 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.
>
> The current ECC-based algorithms using AES key wrap (code -29 through -34) use HKDF with SHA-256, so do not conform to CNSA 1.0.

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	
Confidentiality	1	ECDH-SS + A256KW	-34	

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 ECDH-SS algorithms, optional for ECDH-ES
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 /
  2, / 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 /
  h'82a081c9' / CRC value /
]
```

Figure 6: Primary block CBOR diagnostic

```
[
  1, / type code: payload /
  1, / block num /
  0, / flags /
  2, / CRC type /
  <<"hello">>, / block-type-specific-data /
  h'4ec359d2' / CRC value /
]
```

Figure 7: Target block CBOR diagnostic

Together these form an original bundle without any security operations present. This bundle is encoded as the following 77 octets in base-16:

```
9f890700028201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f42404482a081c98601010002466568656
6c6c6f444ec359d2ff
```

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, h'82a081c9'], / 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 /
  2, / CRC type /
  h'1fd25f64a2ee33e774abe16700bcfd9cf12ea5f7d841', / ciphertext /
  h'47abdef0'
]
```

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: 'ExampleA.1',
    / 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'8201662f2f7372632fa200012001890700028201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
404482a081c901010040', / 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: 'ExampleA.1'
        },
        null, / payload detached /
        h'ec8260a38a1a00fef2cd4aae063f50f01c5645e84c6c4893ca895eed44
ef60a5f50f9adf5cc5654499b881e589637805' / tag /
      ]>>
    ]
  ]
]
```

Figure 14: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 180 octets in base-16:

```
9f890700028201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f42404482a081c9850b03000058608101
03018201662f2f7372632f818205a200012001818182115845844 3a10106a1044a45
78616d706c65412e31f65830ec8260a38a1a00fef2cd4aae063f50f01c5645e84c6c
4893ca895eed44ef60a5f50f9adf5cc5654499b881e5896378058601010002466568
656c6c6f444ec359d2ff
```

A.2. ECC Keypair COSE_Sign1

This is an example of a signature with the signer having a P-384 curve ECC keypair identified by a "kid".

```
[
  { / signing private key /
    / kty / 1: 2, / EC2 /
    / kid / 2: 'ExampleA.2',
    / 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'8201662f2f7372632fa200012001890700028201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
404482a081c901010040', / 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: 'ExampleA.2'
           },
           null, / payload detached /
           h'9c64328dfe9570262f5be687c35cc51ced48b8682d2a61d8baadfd3410
   233634251c2c1862b0a194b8503985931051a77731a74a1514b83092d7c662e6dbcd
   a2629af72b24bf1cc3c5e2552f54ddfcc1762e6bc46fd5e6c2137e4a695e563e
   ae' / signature /
         ]>>
       ]
     ]
   ]
```

Figure 17: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 229 octets in base-16:

```
9f890700028201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f42404482a081c9850b03000058918101
03018201662f2f7372632f818205a200012001818182125876844 4a1013832a1044a
4578616d706c65412e32f658609c64328dfe9570262f5be687c35cc51ced48b8682d
2a61d8baadfd3410233634251c2c1862b0a194b8503985931051a77731a74a1514b8
3092d7c662e6dbcda2629af72b24bf1cc3c5e2552f54ddfcc1762e6bc46fd5e6c213
7e4a695e563eae8601010002466568656c6c6f444ec359d2ff
```

A.3. RSA Keypair COSE_Sign1

This is an example of a signature with the signer having a 3072-bit RSA keypair identified by a "kid".

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: 'ExampleA.3',
    / alg / 3: -38, / PS384 /
    / ops / 4: [1, 2], / sign, verify /
    / n / -1: h'c14d4f1f3ed0913404c7ceffda1bb273e7cd8b575840d03a1048
5f3fead54bc2be84f21a771e56cb3a547db2fb1dd583932e5baa1d755dcaac0aaa78
dffee68f1e187121b22bf965a777a4dc7a7e8633fbc83867caf503d22f3d7f579bad
f3dd706cce0f0855f48eab3d6bd5e0fdef354b1469dd361e9f157e52add65ad9aa38
c281b8d4ef6818670816afcfd1d851e94fc6e70a5fc277c6307e6d68716d137b5d79
4b613b45cd911e59b94454fa70e75c72d9b4cffb9ff496e602a21f09de8f9062eec4
fddc480e5a1f854b18bd412ea0f8ed6f08f63516a4de42afffc94940ef7959363126
40116f7789702bee01a6118a7b6ee5d5496397fa58f408c968157e82a9b3f9579526
c301a9cd012f5c29c829425e581ba474a9a116b5ade9f60fb01fc45b03862d6e6eaa
3f586e456914370953ae725eb1deac8965da2d7a0568fddf4be2325fc2ee3ea4d338
e367e8e5a1da782a6bd5bf0291edb1cbd661c6aa2328a88df575b14b1bed84643a0f
57c1075cabbfcfbc42ef7637300612d5f9ef',
    / e / -2: h'010001',
    / d / -3: h'1ea457800a503bf6fa865aa677d7d479dccb84f9f8c2a174d582
f0c7c19299456037d3e70fc343eff2feff6eb6b19bd89525654b7a0afd035fdd504e
594d0a15b2d2c8baeb885eab0219370d94ef674268a31714964edbbc5f025e798548
ab1e8b0551c429469d935b75764426667ff1109b464d80ed94109a00978fa216dcb8
785636f603936fec2e933b6b1eb12e09903cbaaae17e2d72c1da30bebc884017da11
470fe7e4f5f964a31acec85d16365ebdd2b6aa679160cf90db91abde3fcf7701cbc3
28ad0bc0e7a633700d220dfac58c63e16f6e45e4f936ade468e1b398e52dda883fa6
59b7442af44c9152b487e1217c541125cfc48d75652b087beb8e9198e546a207e125
369f051780de3f8831a5427581050ddaeb3a91d9d1438d6e81288cecc246b533911f
678bfe1accd6bac04093031736b2d7e889d455cef1d7a3fd957c42de8dbf59ad33f3
0ed6b60b83ac9c3f8f2b4c94b2782770ebeed3c5503219729891cb5c26115ba8dc00
56cefe8b2cdb08dc5e1f01c3748476b6d99d',
    / p / -4: h'f5ace2298a583123dbc945ecdb640fb26bfddf00aa23ad065b92
18505bcaf50f736d41025db450ef387d901df5e655c80e08437d4f0caec4f2408bc4
38c76e909f033f10e0cdcc92189c3e22e5172ca443f10510854ebfe753df33712549
166af083ad45027ae03e9b56c2e505611e2dce649f046aa82cc40a0b071bfb8551b9
5070badf994afa4053163454923689ceb41270897c1235019eefb44c3cab49d596a9
ae0f8cfd15f9f795104714f77235fe152adbe846df3462fff61a38c40de5',
    / q / -5: h'c96cf68ed93426255732edf523f3cf54248a6439dbf2d3285ce3
5c74b9211b750997920451f970560f58d12bbad498b5d1a1fec4ec1162c075678816
b4fb1a4aff747871ac55e8792361c2968864ae33dc82299475b5d3b5c6380b1ed64a
56c5ec21cfaa90967aac499daa8ddbe8980e98ef0260c73731488d5ba2ba92d0e8f6
c2cfb6a1367f72858374d2588779efb2e2c1533482a95496a7c5c171c463f71ca8ee
45146f77cebde57be857075a9d71f78116ffc3be1bac428ba1456c5f8b43',
    / dP / -6: h'0649573c32e310d6d70fee6f222a0c50c77ca69130c95aeb17b
```

```
   ac44e821ace6c87ca9ae84197949e9a767412a03135aeb9d5324ce991ec82f3a3fd2
   f97385b36ee2aba1967773caddc5d5b25af71095e66b2ab2b820dc2d15b8f1194ea9
   c552b855e093803d93b15bf09d850ddf35f3f52d1b653f99ab6128a23401a5234562
   404cfeda83d16f312644de426e9dae569d9a7c323717e51c6e9d73e68d9009512171
   9de6f5d6f3879be011d7a8429d4cd56e419c5a8caef793ab34c0bddb9fe95',
       / dQ / -7: h'62bb626fcacfe112d4974644af06c74dbb4b8aad41bed8fa23e
   dde57e896edda84852331b2eccdbfa16e2bb97faecddbf191b24bdc5af948d543965
   56b08da6e80a11a98bd9cae831270ccecf496453d6e8ceeccb29619dc33f92c9a44f
   7d368d8c20a04d532ad96ddcec6d71a3ffca8cb15fcd86b4e067e45abf12bfae3240
   e3097983195810b259eb61895047324a74eb6ec8e04adf3a495403dfe0201ee12c24
   b68d9077a7680668841eec6d007f4e11909a8fccda6cadd238c3d774dadf9',
       / qInv / -8: h'cbd0a9d2d3e1922948906ffa45f27dc75383a81b3fd7fe57e
   ce7f3e9d4bb1b3139696208fccedbeb1f3fc58493af5806fedd4bf496d087012a874
   1bcdeabc590f3810ec77dfb8c38fc3ae68b74c22f6a998c295cd191dfcfe17b029ba
   f7687d6a5a2672231dcb67cb93a854dee715319b195716bad1636382c2e124fcfed2
   eb25be7f3a969cd5ce0f60c88213a5fb9e8de7d99fb54867c3f604925da9f522ca67
   9633b134468882364be6595a55648a41fb56ae658f27ab704055d4c23bb95fa'
     }
   ]
```

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'8201662f2f7372632fa200012001890700028201692f2f6473742f7376638201
   692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
   404482a081c901010040', / 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: 'ExampleA.3'
           },
           null, / payload detached /
           h'687790c647271611102c8baf056046dac4184ee6e4e068d3b01a101723
   9840714dfa5a9ed593680c9415a4dfb1e1473bb7807d9c0d614041b5dfbf963a0ba7
   965cb446ac44602d8e17ebaf888d4a86edec6f47f71ba36f26b0ec657ac73f0edf08
   381e1d2496f782c8c114728bab1e4ab0801531998e13e1ecb39a9e011142cb3b321d
   ecfc08845dbc0685d96ac089df5c09937a8f47c46078d9dbc07725b9a85b85b7c570
   8c6dfbacded9aea48ab492c1188e6b597a0bd81847519683ce5fd315f94edd44bf09
   f842a9be66f281dcf62ccfd6257652d0fc6a86e0bda5132effa84a71c271aa975ac5
   4511a70ddb5a8dd2ab8b9b5fd2680d27a09277d3d7777d0da83dafc1cd97753e35c0
   d7a4b6ea0d2a74d278f39ff365f3ed61d4c50b35e1bb5c23e8c5778d43558f6e1d7f
   9cd8ac38c12d33eb11cd698618f4d5536b1fe3482b42e69baf266bc82a2cfc0577be
   126b6b6aac0134273759b64d3d7512da810092fa6345e26ede9d1a3e20336b2a7448
   58928dd1158b4dc6e8037f4bfba61e' / signature /
         ]>>
       ]
     ]
   ]
```

Figure 20: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 520 octets in base-16:

```
9f890700028201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f42404482a081c9850b0300005901b381
0103018201662f2f7372632f818205a200012001818182125901978444a1013825a1
044a4578616d706c65412e33f6590180687790c647271611102c8baf056046dac418
4ee6e4e068d3b01a1017239840714dfa5a9ed593680c9415a4dfb1e1473bb7807d9c
0d614041b5dfbf963a0ba7965cb446ac44602d8e17ebaf888d4a86edec6f47f71ba3
6f26b0ec657ac73f0edf08381e1d2496f782c8c114728bab1e4ab0801531998e13e1
ecb39a9e011142cb3b321decfc08845dbc0685d96ac089df5c09937a8f47c46078d9
dbc07725b9a85b85b7c5708c6dfbacded9aea48ab492c1188e6b597a0bd818475196
83ce5fd315f94edd44bf09f842a9be66f281dcf62ccfd6257652d0fc6a86e0bda513
2effa84a71c271aa975ac54511a70ddb5a8dd2ab8b9b5fd2680d27a09277d3d7777d
0da83dafc1cd97753e35c0d7a4b6ea0d2a74d278f39ff365f3ed61d4c50b35e1bb5c
23e8c5778d43558f6e1d7f9cd8ac38c12d33eb11cd698618f4d5536b1fe3482b42e6
9baf266bc82a2cfc0577be126b6b6aac0134273759b64d3d7512da810092fa6345e2
6ede9d1a3e20336b2a744858928dd1158b4dc6e8037f4bfba61e8601010002466568
656c6c6f444ec359d2ff
```

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: 'ExampleA.4',
    / 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'8201662f2f7372632fa200012001890700028201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
404482a081c901010040' / 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: 'ExampleA.4',
          / partial iv / 6: h'484a'
        },
        null / payload detached /
      ]>>
    ]
  ]
]
```

Figure 23: Abstract Security Block CBOR diagnostic

```
[
  1, / type code: payload /
  1, / block num /
  0, / flags /
  2, / CRC type /
  h'1fd25f64a2eee2ff1a1ab29812ba221874380974c13b', / ciphertext /
  h'2086c017'
]
```

Figure 24: Encrypted Target block CBOR diagnostic

The final bundle is encoded as the following 149 octets in base-16:

```
9f890700028201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f42404482a081c9850c03000058318101
03018201662f2f7372632f818205a200012001818182105783 43a10103a2044a4578
616d706c65412e340642484af68601010002561fd25f64a2eee2ff1a1ab29812ba22
1874380974c13b442086c017ff
```

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: 'ExampleA.5',
    / 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'8201662f2f7372632fa200012001890700028201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
404482a081c901010040' / 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: 'ExampleA.5'
            },
            h'917f2045e1169502756252bf119a94cdac6a9d8944245b5a9a26d4
03a6331159e3d691a708e9984d' / key-wrapped /
          ]
        ]
      ]>>
    ]
  ]
]
```

Figure 27: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 209 octets in base-16:

```
9f890700028201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f42404482a081c9850c030000586d8101
03018201662f2f7372632f818205a200012001818182186058518443a10103a1054c
6f3093eba5d85143c3dc484af6818340a20124044a4578616d706c65412e35582891
7f2045e1169502756252bf119a94cdac6a9d8944245b5a9a26d403a6331159e3d691
a708e9984d8601010002561fd25f64a2ee33e774abe16700bcfd9cf12ea5f7d84144
47abdef0ff
```

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: 'ExampleA.6',
    / alg / 3: -11, / direct+HKDF-SHA-512 /
    / ops / 4: [7], / derive key /
    / k / -1: h'6c4e5271e211e0c8329ab8f363097f16516a459f12a4060cf016
4968fdccbd63'
  },
  { / derived CEK /
    / kty / 1: 4, / symmetric /
    / alg / 3: 3, / A256GCM /
    / k / -1: h'a2db6ff560dfd645999362dc1d39eff1d77336c93e973baf5053
f4733c19ade4'
  }
]
```

Figure 28: Example Keys

The external_aad is the encoded data from Figure 10.

```
[
  "Encrypt", / context /
  h'a10103', / protected /
  h'8201662f2f7372632fa200012001890700028201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
404482a081c901010040' / 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: 'ExampleA.6',
              / salt / -20: h'2fa8c8352aea17faf7407271a5e90eb8'
            },
            h'' / empty /
          ]
        ]
      ]>>
    ]
  ]
]
```

Figure 30: Abstract Security Block CBOR diagnostic

```
[
  1, / type code: payload /
  1, / block num /
  0, / flags /
  2, / CRC type /
  h'cd3367b96d5b3f493cdc117f9a392f02d6d4b93f6726', / ciphertext /
  h'1524984b'
]
```

Figure 31: Encrypted Target block CBOR diagnostic

The final bundle is encoded as the following 187 octets in base-16:

```
9f890700028201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f42404482a081c9850c03000058578101
03018201662f2f7372632f818205a200012001818181821860583b8443a10103a1054c
6f3093eba5d85143c3dc484af6818343a1012aa2044a4578616d706c65412e363350
2fa8c8352aea17faf7407271a5e90eb840860101000256cd3367b96d5b3f493cdc11
7f9a392f02d6d4b93f6726441524984bff
```

A.7. ECC Keypair COSE_Encrypt with Key Wrap

This is an example of an encryption with an P-384 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: 'ExampleA.7',
    / 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'8201662f2f7372632fa200012001890700028201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
404482a081c901010040' / 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: 'ExampleA.7',
                 / 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 319 octets in base-16:

```
9f890700028201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f42404482a081c9850c03000058db8101
03018201662f2f7372632f818205a200012001818182186058bf8443a10103a1054c
6f3093eba5d85143c3dc484af6818344a101381ea2044a4578616d706c65412e3720
a4010220022158302f88f095c45c96e377e18d717a5e6007ce8f6076ae82009d1637
5e1b9abaa9497a4bde513be6c9b0e7dae96033968c45225830fd27656fbb97f789d6
67f40d73b65ab362b22dd23bf492bee72bf3409f68dddf208040a5fcbcbee7454574
1e2866cb2d58280eaff015e61418d8910ba25ed9733450558b6a20ab410f3c925b01
ac8d3aefcc12433f9563da401d8601010002561fd25f64a2ee33e774abe16700bcfd
9cf12ea5f7d8414447abdef0ff
```

A.8. ECC Keypair COSE_Encrypt with HKDF

This is an example of an encryption with an P-384 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: 'SenderA.8',
    / 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: 'ExampleA.8',
    / 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'8201662f2f7372632fa200012001890700028201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
404482a081c901010040' / 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: 'ExampleA.8',
              / sender kid / -3: 'SenderA.8',
              / salt / -20: h'2fa8c8352aea17faf7407271a5e90eb8'
            },
            h'' / empty /
          ]
        ]
      ]>>
    ]
  ]
]
```

Figure 37: Abstract Security Block CBOR diagnostic

```
[
  1, / type code: payload /
  1, / block num /
  0, / flags /
  2, / CRC type /
  h'925c33206eb89fdc076f17489de3056e997a227b8d53', / ciphertext /
  h'8eec52c4'
]
```

Figure 38: Encrypted Target block CBOR diagnostic

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

```
9f890700028201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f42404482a081c9850c03000058638101
03018201662f2f7372632f818205a200012001818182186058478443a10103a1054c
6f3093eba5d85143c3dc484af6818344a101381ba3044a4578616d706c65412e3822
4953656e646572412e3833502fa8c8352aea17faf7407271a5e90eb8408601010002
56925c33206eb89fdc076f17489de3056e997a227b8d53448eec52c4ff
```

A.9. RSA Keypair COSE_Encrypt

This is an example of an encryption 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 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: 'ExampleA.9',
    / alg / 3: -42, / RSAES-OAEP w SHA-512 /
    / ops / 4: [5, 6], / wrap, unwrap /
    / n / -1: h'bb4917794481770c92a1ba6a35fbe0677a5c3669cd39c530985a
234765d0c0acc874925b1578e08f5d71dec62c1d28bb237fc3f1ddf8f01cab5ac207
5ade1747958d818fd332781891dbda85e00d0006a538f88d28900f69d93c340bd7da
8d47d0e63b448671b885d35a275a7204ed15bea0276ace4bbca291d2843b4454fce8
5faf78056753b6331b01f54c52eca23c0c255ea53919a972b548777049dc64bc4261
7ae74fc1af5bd10d72102f32347e12161d9fb1d43c9cbf26a49bd65a6b282276a634
15c52b36ce2a186f0ecc6b15a4c596c67a9eafca72e665c3a91062b22d1f00d05fb3
fb120f34263406c64848d93baa65985a7974aafc39f83a39c896c907da9b7e6df1a6
f9c3588ebd5ae5d6dfce569e15d17a4594098c1606b3b94cfdeff8dc41e56e9592fc
59de96b6aae1729444ee28e6fedd59e432f0670465a65212774ece52c205748ec207
db332feef700d2b4a2c2a7d40efddac627d816b872c6e12b074704b12f2dbb92b44f
7bd799a2848ef0c17e1783baa33e89c1bb4b',
```

```
      / e / -2: h'010001',
      / d / -3: h'8d0b34532ce688fadcb4dea67fd303ad0c84632f87d2cf57e59a
    80319defb97dfafa13c247d3828c6bcac2567507108e84ad8937cd25676ac70f45d6
    07360efa5efd3daf42a19758dfd557775b56da4b68bc4f70c728ef09df397b57e01e
    17f2c96afba541d096365e9c549df5ed82d9d9c0d43ca3f454af1c6701afd1749636
    03f20f52f647225a24e81403c72dd0336ff99027d44f12b073d87faa8c263f1fe505
    03757be3210c455df6e92f9aaf89a63ec49b884af7648c168a7116848087b94db5a8
    2435e98249723543fdbe8bf420faa6f578c382738a2a2753e7886e8152ba5ec8291d
    002b87a068a73fc5f3a3379424582d1ed5b4c338475c8de509f37c3092d3fd8337b0
    9b0d9725add3380d921d4f9f90700116b5543cb8a40c3ec0e4661cf09f0ebf62c57c
    dbf63c59390d6f1d2dbd2ea09be5c21d2732109e7787cb9a4582d8c2be712a2d9355
    c1b8ba1e597dc2012bb920e551a6fddc0c7db08ab32b0add6ddedab6b70b4c3105dc
    ed09a49c6cf6e325b8b80c65fc1859fcd5',
      / p / -4: h'fa214874981ce573589c4eb4682c12aed490c66714a4e339ea2d
    b376b6dac4bd997fdeacccd4b514daeda487b86a273dec8746a5debb3f776c46367c
    f163f968c76900de21a20b75201b9a376327158e90a52e3e24e3c60b79102a572ad9
    f859364fdce1c14da0379480ee87c20fd54454847a41c644fff9e9e72b6d42dbcd5b
    7d343abbf785e72d494fd60e309322e5bcb20763f56c6000ae975eb6d4c23e1f3e0b
    6f6d52b74cefaa6045fbd0697740895b45af918faf75febec37f6e88eb55',
      / q / -5: h'bfae414a486903f3f203382d3995dcae8e1e716b8835d1126819
    4879d9dad3d57396e3fd52a16272221d25a2f8e82eccc29c16751061e903566825cd
    66e562bad038b002684356411bc323d8212c8b7aac4dd481b511e9de45ab3b6cab78
    50d30f2861e0e7c6778d26b19458fff4f74d2b65af87234a090ab241ea8a51b8cb15
    294b1b283bead83f9064cb32cbe0f25807ee946484c6a777c19a7bd2a214cbc9ed17
    8552e0afd7748511333375753852fb0b4e9c8d4fcab2d2372be59c104c1f',
      / dP / -6: h'92f19ca44a7ca75b751216b6ab8040d58eb122ad8a16381b5cf
    4ce3a8ebfc4d6f1e78a04902ce1d8c7a8d68099195bc6683f2c84e36db3a24fec8bb
    42907a78d23a10f4e7009c79b5e6a78d5d31d31efd8100233a5ee5df97d7cbeb308c
    c96b6aa4e8e9fddb4e1cbe5253d7c69c86d6cc00e37d88e4718ee53b867edbf5a6bb
    134c3cb4183ef995924798f72349d2be235518d3feefd6504e18cb1aacd20f3e7dcc
    65106b39255d3728f2e6dfa090b72d17eda5883361b4941880647c5c31025',
      / dQ / -7: h'933ea1191716d4da8886c098bd2bca22ad39e596dd43ba1f91a
    81a6cc055c174af1eb274df0cea3b12c9a127d85d43d6378900175d4659611ef7525
    2bf4066df6b24a0d0b89741a332586d2892134df2267a834c40744a5b5cd97504bd9
    3e742bada22964a75c350c2f0972ce7329ee6c0f79427138cc3f55b8a1749ba0d62b
    416cc83481cff02af91945c23e14a23e04bf79236c568752d21a4328a53c7f5e4602
    5395db90c5b4e3f0a3f72c04013cc6adcfcbe762f5d5e90eda0e2f947ebb1',
      / qInv / -8: h'2f61ebed182ff0375be59300f2f0f4302f915274756b13dfa
    3847b56259c87a204e7188656460afec04bf8889ad2ab6cd54d56cbff63eeac06620
    ec6cadca22ba4cc4ee29b6195aaab25ef33455ef204eb75f93e9fc2b0c7bfe11f112
    7c2b9102e729a504eb1bd350c70568acbab5b5feffa8272f0458ba66491fd93387e8
    6b8c8c2ed69845b6dffc0b3800dc175d3bdf40e154053141e54db17f9515dfa719de
    b426775bac26854b539e18176f89e785bacd4672534f683f80b2cc7927bf8f7'
    },
    { / 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'8201662f2f7372632fa200012001890700028201692f2f6473742f7376638201
692f2f7372632f7376638201662f2f7372632f821b000000bd51281400001a000f42
404482a081c901010040' / 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: 'ExampleA.9'
               },
               h'50901651a7f2d911da19ced267bf2390bd9af7d0e0617a3212c59c
   f1ae237041aa81ff8e169c49a570be2c5eeced21c4666d4b385b36462e486f011791
   ec7f86e9b0afe0affafc12f26d605ef13396675d6d4642448a5fd9a1cfdd999f1423
   31d894501f8b82d08e7d1703ab14eaf510bcc4e18e373ab2ed502ebb99dc0035f393
   c4cbdea8b40535e528017087ee700442539e7cf079950de91c0aa9f058c66e15a640
   eba39b4e619c4daf6c08beaf654932f8f88ad8685e87402f75be68bf3dd2e5539a7d
   0ea880ec89788c36dc3a6603eda6999f519eed0f62302ea92adc13d52bf7898eb1ab
   1aa587bf8f278059ede7c75204d3d69f67b00b50cd70a8724eb2a204c275981af92a
   4ae21b77d9ca8be275fb4a1edbba3edcae4a0f964ca913326a9507c4c6647adc9487
   82136036f73cbfba33e1b5977591931b99ce536015bfb89c062c3208189bdc43e530
   6cdaefa81a769df267d00233e375e5b0b027974fda218f318c7cd7c1fdbfe8548fb2
   71f3b14de9a50d7bb23e26feb3cc1ea882' / key-encapsulation /
             ]
           ]
         ]>>
       ]
     ]
   ]
```

Figure 41: Abstract Security Block CBOR diagnostic

The final bundle is encoded as the following 557 octets in base-16:

```
9f890700028201692f2f6473742f7376638201692f2f7372632f7376638201662f2f
7372632f821b000000bd51281400001a000f42404482a081c9850c0300005901c881
0103018201662f2f7372632f818205a200012001818218605901ab8443a10103a1
054c6f3093eba5d85143c3dc484af6818340a2013829044a4578616d706c65412e39
59018050901651a7f2d911da19ced267bf2390bd9af7d0e0617a3212c59cf1ae2370
41aa81ff8e169c49a570be2c5eeced21c4666d4b385b36462e486f011791ec7f86e9
b0afe0affafc12f26d605ef13396675d6d4642448a5fd9a1cfdd999f142331d89450
1f8b82d08e7d1703ab14eaf510bcc4e18e373ab2ed502ebb99dc0035f393c4cbdea8
b40535e528017087ee700442539e7cf079950de91c0aa9f058c66e15a640eba39b4e
619c4daf6c08beaf654932f8f88ad8685e87402f75be68bf3dd2e5539a7d0ea880ec
89788c36dc3a6603eda6999f519eed0f62302ea92adc13d52bf7898eb1ab1aa587bf
8f278059ede7c75204d3d69f67b00b50cd70a8724eb2a204c275981af92a4ae21b77
d9ca8be275fb4a1edbba3edcae4a0f964ca913326a9507c4c6647adc948782136036
f73cbfba33e1b5977591931b99ce536015bfb89c062c3208189bdc43e5306cdaefa8
1a769df267d00233e375e5b0b027974fda218f318c7cd7c1fdbfe8548fb271f3b14d
e9a50d7bb23e26feb3cc1ea8828601010002561fd25f64a2ee33e774abe16700bcfd
9cf12ea5f7d8414447abdef0ff
```

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 and Appendix A.8. 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