Delay-Tolerant Networking
Internet-Draft
Intended status: Standards Track
Expires: 6 December 2025

B. Sipos
JHU/APL

4 June 2025

Bundle Protocol (BP) Security Associations with Few Exchanges (SAFE)
draft-sipos-dtn-bp-safe-00

Abstract

This document defines a protocol for negotiating scoped security associations between Bundle Protocol version 7 (BPv7) agents within a delay-tolerant network (DTN). Security associations are used to amortize the costs of asymmetric-keyed security operations and allow for efficient and high-throughput BPv7 security within a public key infrastructure.

Status of This Memo

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

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

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

This Internet-Draft will expire on 6 December 2025.

Table of Contents

1. Introduction

The combination of Bundle Protocol version 7 (BPv7) [RFC9171] and Bundle Protocol Security (BPSec) [RFC9172] enables security to be applied at a fine-grained level to individual blocks of a bundle.

When operating within a Public Key Infrastructure Using X.509 (PKIX) [RFC5280] environment, in the absence of any kind of online protocol between the security source and expected acceptor(s) there are two extreme alternatives for the use of public keys for integrity and/or confidentiality:

1. The security source can use a public key of the security source directly for signing each target or use a public key of the security acceptor directly for each target (for either key agreement or key encapsulation).

   This is the strategy taken by the email security of S/MIME [RFC8551] and asymmetric algorithms of CBOR Object Signing and Encryption (COSE) [RFC9052]. While it ensures that each security operation can be processed independently it also introduces a large overhead because asymmetric-keyed algorithms are likely to be orders of magnitude more resource intensive than symmetric-keyed ones.

2. For key types which support key exchange, such as elliptic curve (EC) keys, the security source can use a public key from both the security source and acceptor to perform a one-time key derivation of a shared secret, and from that secret a pseudorandom function (PRF) can be used to derive symmetric keys known to both entities.

   This allows the cost of one-time key exchange and PRF operations to be amortized across all of the cryptographic operations using the derived symmetric keys. But without any additional scoping added to the derived keys (_e.g._, valid time of use or volume of data processed, restriction to specific ciphersuites or algorithms, etc.), the keys themselves will be vulnerable to overuse or cross-use vulnerabilities.

   This technique alone also provides confidentiality of the derived symmetric keys but does not intrinsically provide any authentication of the peer entity (via some identity binding to the associated private key). It also does not allow for forward secrecy because the original asymmetric keys need to be long-lived enough to handle all communications between the entities.

In order to both amortize the costs of asymmetric-keyed algorithms and to provide a separate means of mutually authenticating a peer BP node, including a proof-of-possession for the private key associated with a known public key, this document defines the Security Associations with Few Exchanges (SAFE) protocol.

The SAFE protocol operation is explained in detail in Section 2. It operates as an "in-band" application over BPv7 as depicted in Figure 1. This is similar to how the Internet Key Exchange Version 2 (IKEv2) protocol of [RFC7296] operates as an application over UDP/IP.

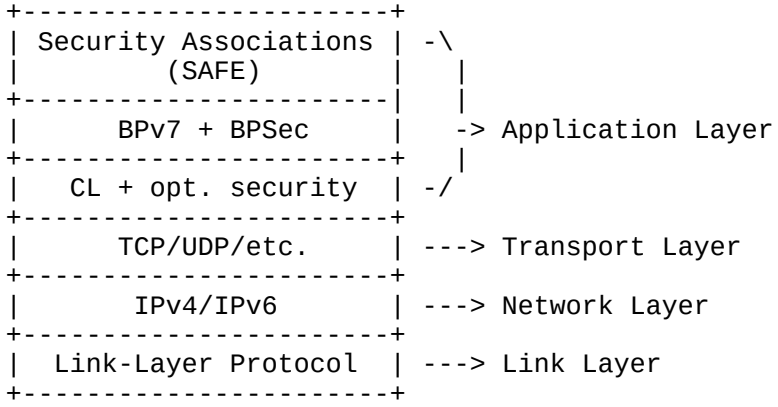


Figure 1: The Locations of SAFE and BP above the Internet Protocol Stack

1.1. Scope

This document describes the format of the protocol data units passed between BP nodes for security association negotiation and defines behavior at message source and destination nodes. It also defines how each participating node acts on those security associations to process BPSec security operations.

This document does not address:

* The format of protocol data units of the Bundle Protocol, as those are defined elsewhere in [RFC9171]. This includes the concept of bundle fragmentation or bundle encapsulation.

* Logic for routing bundles along a path toward a bundle’s endpoint.

* Uses of security associations outside of the use cases explained in Section 1.2 or when combined with other techniques such as bundle-in-bundle encapsulation (BIBE) [I-D.ietf-dtn-bibect].

* Policies or mechanisms for using BP extension blocks for purposes not defined in this document. Some networks could require specific extension blocks to be present for valid traffic.

* 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.

1.2. Use Cases Considered

Based on terminology from Section 3.1 of [RFC9171], the four data plane interaction points between a BP Agent (BPA) and other entities are shown for two agents in Figure 2. For simplicity that diagram shows a situation where there is reachability between the nodes in one direction, but typically reachability between nodes would be in both directions (via similar or dissimilar convergence layer types and/or hop counts).

Two of the interaction points are with application(s) registered as endpoints in the BPA (transmit and deliver) and two are with underlying convergence layer(s) used to transport bundles between BPAs (forward and receive) for each hop. Within a BPA there is logic about how and when to deliver, forward, retain, delete, discard, or some combination of those actions which are defined in Section 5 of [RFC9171]. This logic is indicated in later diagrams by the "(Decide)" label inside a BPA block.

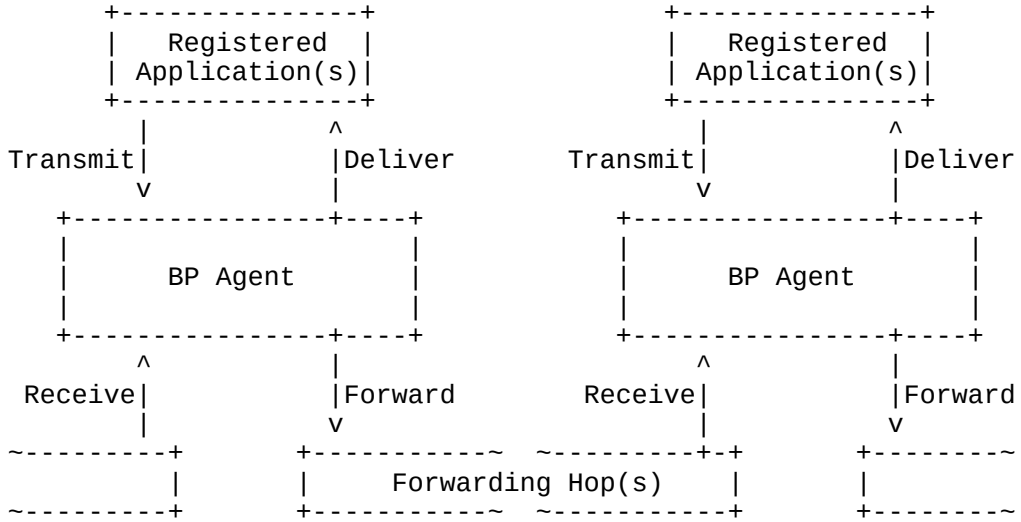


Figure 2: BP Agent data plane interaction points

This document considers two principal use cases to narrow the scope of discussion, each described in the following subsections. More complex use cases can be achieved by this protocol, either by more complex interaction with a BPSec entity or the use of techniques such as BIBE of [I-D.ietf-dtn-bibect] to perform secure tunneling between pairs of security gateway nodes.

1.2.1. End-to-End Security

The end-to-end use case is where the SAFE entities negotiate secondary SAs which match endpoints on the participating nodes themselves. This use case corresponds to the end-to-end mode of Security Mode Selector (SMS).

For this mode the bundle flows and security processing will look like what is depicted in Figure 3. The negotiated security service is sourced immediately upon bundle creation (after the corresponding ADU is transmitted by the source endpoint), has a lifetime equal to the bundle itself, and is accepted immediately before bundle delivery (of the ADU to the destination endpoint).

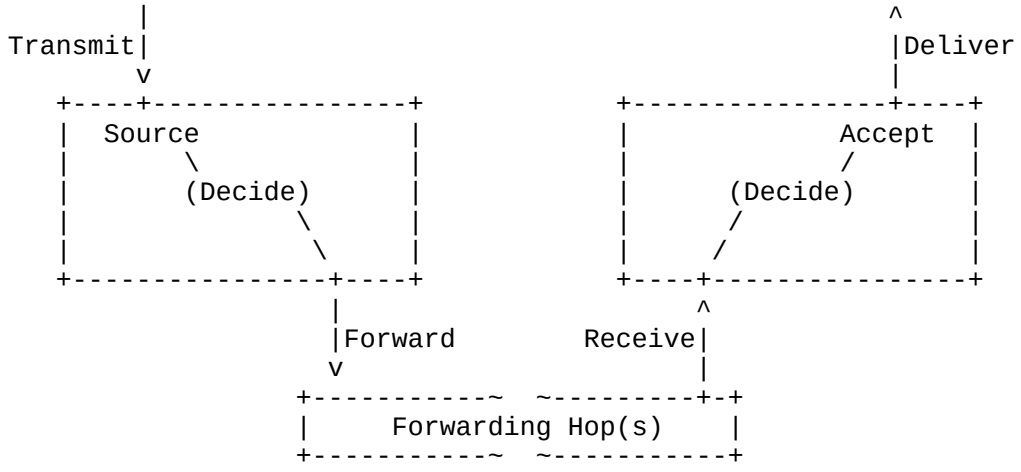


Figure 3: End-to-End security flows

The purpose of this mode is to ensure that specific traffic has integrity or confidentiality protection for its entire lifetime, applied as close to the endpoints as possible. This protection also includes any possible storage at the source and destination nodes.

1.2.2. One-Hop Security

The one-hop use case is where the SAFE entities negotiate secondary SAs which match arbitrary endpoints but apply to bundles traversing a single hop with a neighbor node. This use case corresponds to the one-hop mode of Security Mode Selector (SMS).

For this mode the bundle flows and security processing will look like what is depicted in Figure 4. The negotiated security service is sourced immediately before forwarding (specifically to the peer node of the SA), has a lifetime of just a single bundle hop, and is accepted immediately upon reception at that peer node.

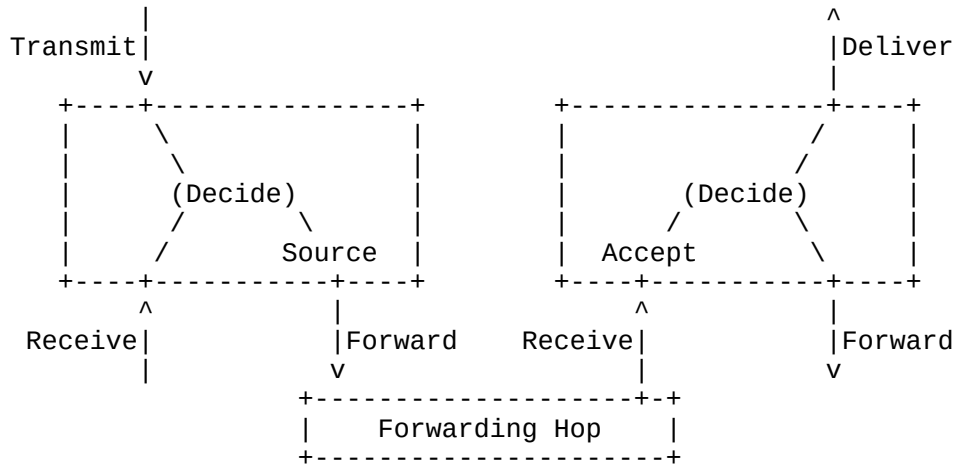


Figure 4: One-Hop security flows

The purpose of this mode is to supplement security mechanisms (if any) provided by the forwarding convergence layer adapter (CLA). This allows the receiving node to authenticate data from the previous node during reception, by transitively linking the one-hop security back to the mutual authentication and proof-of-possession from the Initial Authentication (IA) activity.

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 following XPath expression:

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

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

```
start = safe-pdu-seq / safe-msg / safe-msg-bstr
```

1.4. Terminology

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.

Terminology used within the SAFE protocol includes the following:

SAFE Bundle: A bundle with a source or destination EID having a well-known service identifier allocated for SAFE (see Section 12).

Participating node: A BP node which sources and/or delivers SAFE Bundles via a BP Agent.

Interaction point (of a BP Agent): One of the four locations on the data plane by which a BP Agent interacts with other entities in a BP node.

Security Association (SA): An agreed state between two participating nodes which contains the result of an authenticated key exchange and some number of derived symmetric session keys.

Activity: Each SAFE activity consists of a sequence of data being exchanged, via protocol messaging, between two nodes to achieve a specific narrow goal.

Message: Each step of an activity is a message containing a specific set of data items. Each message has a deterministic binary encoding using CBOR. Some messages are constructed as responses to earlier messages in an activity and need to be interpreted in the context of the whole activity.

Protocol Data Unit (PDU): A sequence of messages are aggregated together into a single protocol data unit to be exchanged with the SAFE entity of a participating peer. All SAFE PDUs have some kind of application-level confidentiality of the contained messages.

Application Data Unit (ADU): When supplied to a BP Agent, a SAFE PDU is handled as the ADU of a bundle.

Endpoint Selector (ES): A filter for BP traffic based on specific fields of the bundle primary block or well-known extension block types.

2. General Protocol Description

The service of this protocol is the establishment and management of one or more secondary security association (SA) between two participating peer BP nodes (that may or may not be BP neighbors) which are used to inform a BPSec entity on each node how to secure specific traffic.

The initial confidential channel with forward secrecy, mutual authentication, and proof-of-possession is provided by an Ephemeral Diffie-Hellman Over COSE (EDHOC) session [RFC9528]. The EDHOC session also allows using external authorization data (EAD) items to confidentially transport SAND messages.

A side effect of the EDHOC session is the establishment of a primary SA between the two peers. The primary SA provides message confidentiality after the initial authentication EDHOC session. After a primary SA is established, this protocol allows creating secondary SAs through a single 1.5 round-trip activity without re-authenticating. It also allows managing established SAs, including symmetric key decommissioning and roll-over, in order to bound keys over time and over volume of processed data.

The concepts and procedures of SAFE are similar in both form and function to the IP-level IKEv2 of [RFC7296] and MAC-level Port-Based Network Access Control (BPNAC) of [802.1X]. These earlier protocols assume a low enough latency that chaining two-way exchanges over time is not a resource burden or source of major delay, and that if retransmissions are needed they can easily be performed at the exchange initiator side. Because SAFE is expected to operate in environments where one-way latency can be significant (see [RFC4838]), its structure in Section 4 is organized to avoid a sequence-of-exchanges pattern.

Instead of a two-way request--response pattern where all retransmission occurs from the requesting entity, the SAFE pattern is an activity sequence where the final message is purely acknowledgement that the activity has finished. SAFE separates its messaging sub-layer from its packetization sub-layer (see Section 4) to allow messages from multiple simultaneous activities to be aggregated together into a single protocol data unit (PDU). For an individual activity sequence this is results in more messages than two-way exchanges, but it enables pipelining of messages and retransmission from either side of an activity conversation.

2.1. Relationship to EDHOC

This protocol embeds EDHOC as the messaging structure and behavior of the Initial Authentication (IA) activity type. Because the IA activity is the very beginning of a conversation between two SAFE entities, the packetization behavior during the IA activity is a special case containing a single EDHOC message in each PDU. This also means that during the IA activity EDHOC is responsible for confidentiality of SAFE data. After the IA has finished and a primary SA is established, confidentiality is provided by an application-layer use of COSE encryption (see Section 4.4).

2.2. Relationship to BPSec

The reason for establishing SAs in the first place is to be able to provide symmetric session keys to be used by BPSec for security operations on individual target blocks within specific bundles as described in Section 1.2. Part of the scoping of each SA includes a BPSec security context identifier and its options for which each derived symmetric key is authorized to be used.

A secondary SA is scoped by the following aspects which inform BPSec policy on each peer node.

Security Mode:
: This determines when and where security operations are sourced and accepted within the peers.

Validity Time Interval:
: This bounds the SA to an interval of time.

Endpoint Selectors:
: This is an EID pattern used to filter by the source and destination of bundle traffic.

Security Operation Selectors:
: This determines the security service (integrity or confidentiality) and target block types to secure.

Key Use Selectors:
: This is a specific BPSec security context and options for using that context.

2.3. Credential Sources

In the typical use of IKEv2 [RFC7296], TLS [RFC8446], or DTLS [RFC9147] the full credentials (and in the case of PKIX certificates, full certificate chains) are carried in the protocol exchanges. Because the original target for EDHOC was constrained networks, the credentials themselves are purposefully omitted from the EDHOC messages and instead only credential identifiers are exchanged Section 3.5.3 of [RFC9528].

Part of the configuration for each SAFE entity is a set of peer information (see Section 3.1), each of which contains a set of validated credentials and their root-of-trust for that peer. It is still possible for SAFE entities to provide actual credential data, which itself is kept confidential as part of the secure channel established by EDHOC.

2.4. Extensibility

The protocol defined in this document defines a basic set of modes and data types which are expected to be suitable for many bundle security use cases. But the protocol uses extensible IANA registries (see Section 12.3) which allow future specifications to define additional well-known code points. And each registry has a reserved block to allow private networks to make use of private code points tailored to their specific needs.

3. Information Bases

BP SAFE operates by using an activity sequence of messages (see Section 5) to establish and maintain security associations between pairs of participating nodes.

There are two logical tiers of SA, primary and secondary, which are treated here as two separate information tables. How these are mapped to an actual internal data model is an implementation detail, as well as whether an entity treats all SAs together into one pool or separates primary and secondary SA data as indicated in this section.

All private asymmetric key material and all derived PRK and symmetric key material is expected to be maintained outside of the SAFE entity in some form of trusted execution environment (TEE). The key material is included in these information bases as a logical placeholder and to show its association with the other fields.

3.1. Participating Peers

In order to set appropriate retransmission timers, the local entity needs to know expected timing information for each participating peer node.

Name	Description
Peer EID	The transport EID for the SAFE entity on the peer node.
Round-Trip Time	The expected full round-trip time between a sent message and an acknowledgement. This value includes actual one-way-light time (OWLT) of links as well as expected queuing and processing delays.
Acceptable EDHOC Cipher Suites	A priority list of code points from the "EDHOC Cipher Suites" registry of [IANA-EDHOC] which are acceptable to use for EDHOC sessions with this peer. This value is sent in Message 1 when acting as an EDHOC initiator and validated when acting as an EDHOC responder.
TX Credential Types	A priority list of acceptable COSE credential types for EDHOC authentication of this node to the peer. Details on this complex field are described below.
Acceptable RX Credential Types	A priority list of acceptable COSE credential types for EDHOC authentication of the peer node. Details on this complex field are described below.

Table 1: Participating Peer Columns

For the two stores of credential types in the Participating Peers information base, the detailed information present consists of an ordered list of entries, each containing the following.

Credential Type:
One of the "COSE Header Parameters" registry values from [IANA-COSE] which identifies a credential type:

* c5t (22), c5u (23), c5c (25) to identify a C509 certificate following the profile of Section 5.1.2

* x5t (34), x5u (35), x5chain (33) to identify an X509 certificate following the profile of Section 5.1.2
* kid (4) to identify a pre-shared symmetric key (PSK) // leave this in?

Trust Anchors:
One or more type-specific root authority credentials to use as trust anchors for validating end-entity credentials.

Validated Credentials:
A set of credentials which has already been validated in accordance with the profile in Section 5.1.2 against the trust anchors in this entry. Each of these credentials is supplied from outside of the SAFE entity. For TX credentials, this represents local identity configuration for the peer. For RX credentials, this is from peer discovery or pre-agreement with the peer.

3.2. Activity States

Each SAFE entity maintains a table of in-progress activities and their associated metadata, with logical columns as indicated in Table 2.

Name	Description
Initiator	The transport EID for the initiator of the activity. This may be a local endpoint or remote.
Responder	The transport EID for the responder of the activity. This will be the opposite site of conversation from the initiator endpoint.
Activity Index	The index, unique to the initiator, for the activity.
Last Step Transmitted (LTX)	The step of the activity which is associated with the last message sent to the peer.
Last Step Received (LRX)	The step of the activity which is associated with the last message received from the peer. This value is optional for activities initiated by the local entity.

Table 2: Activity State Columns

While the last received step is less than the last sent step, it means that the local entity is waiting for an acknowledging message. After the final message of an activity is received, the associated table row is removed as there is no need to maintain long-term bookkeeping of finished activities.
// TBD about row removal timer and ignoring late duplicate messages.

3.3. Primary Security Associations

Each primary SA allows SAFE entities to provide PDU-level confidentiality and is used as the source of a shared secret from which secondary SAs can be derived. The state of each primary SA known to the local entity has logical columns as indicated in Table 3. The key information of a primary SA will contain exactly one key for each direction
// TBD. The primary SA has no specific endpoint selectors because each is used for security between the SAFE entities themselves rather than any other traffic.

Name	Description
Local SAI	The locally-generated SA identifier for this entry.
Peer EID	The transport EID for the SAFE entity on the peer node.
Peer SAI	The peer-generated SA identifier for this entry.
AEAD Algorithm	The "application AEAD algorithm" from the EDHOC cipher suite negotiated as part of the IA activity. This applies to SAFE confidential PDU processing defined in Section 9.1.
Hash Algorithm	The "application hash algorithm" from the EDHOC cipher suite negotiated as part of the IA activity. This applies to the SAFE_KDF function defined in Section 8.3.
TX Key Information	The primary SA has a single symmetric key for outgoing PDUs to the Peer EID, as a COSE key with contents as defined in Section 8.1. The associated key material is derived from the shared secret created during an Initial Authentication (IA) activity.
RX Key Information	The primary SA has a single symmetric key for incoming PDUs from the Peer EID, as a COSE key with contents as defined in Section 8.1. The associated key material is derived from the shared secret created during an Initial Authentication (IA) activity.
Primary pseudo-random key (PRK)	Internal key material derived from the EDHOC session for this SA, as the byte string PRK_SA1 defined in Section 8.1. This is used to derive further values for secondary SAs as defined in Section 8.3 and context-specific key derivations.

Table 3: Primary Security Association Columns

3.4. Secondary Security Associations

Each secondary SA allows SAFE entities to provide key material and associated policy configuration to a BPSec entity on the same BP node. The state of each secondary SA known to the local entity has logical columns as indicated in Table 4.

Name	Description
Parent SA	The primary SA (Table 3) from which this secondary SA was derived, which includes the Peer EID of the other SAFE entity.
Local SAI	The locally-generated SA identifier for this entry, as defined in Section 6.5.
Peer SAI	The peer-generated SA identifier for this entry, as defined in Section 6.5.
Security Mode Selector	This is a mode selector for this SA, as defined in Section 6.8.
Validity Time Interval	An optional interval of time during which the SA is valid. The logic is consistent with the node time interval (NTI) of Section 6.9. If absent, the SA is valid for all time.
Local Endpoint Selectors	This is an unordered set of endpoint selector items for the local side of the SA, as described below and in Section 6.10.
Peer Endpoint Selectors	This is an unordered set of endpoint selector items for the local side of the SA, as described below and in Section 6.10.
Security Operation Selector	This is a combination of information derived from the negotiated Security Operation Selectors (SOS) during the Section 5.3 activity.
BPSec Context ID	This is a single code point from the "BPSec Security Context Identifiers" registry at [IANA-BUNDLE], which restricts the scope in which the key can be used and provides context for the Key Use Selector details below.
Key Use	This field represents a set of context-specific

Options	options which determine exactly how the SA and its keys are to be used by the BPSec entity on this node. The options are negotiated during SA Creation (SC) as defined in Section 6.12 and per-context specifications (see Section 9).
Key Information	This field represent a set of context-specific key material and derived options for each direction between the two peers. The key material are derived from the shared secret created during SA Creation (SC) as defined in Section 8.3 and per-context specifications (see Section 9).

Table 4: Secondary Security Association Columns

Each endpoint selector item functions as a filter for the BP Agent to determine to which bundles the SA applies. The order in which endpoint selectors and other filters are applied to filter bundles for security processing is an implementation matter and can be optimized to, for example, process the less expensive checks first to reduce the average expense of matching. An implementation is also free to perform additional indexing of endpoint selectors across multiple SAs to reduce total processing expense.

Each key use option contains fields necessary to restrict when and how the key can be used for BPSec security operations. The fields of each item roughly correspond with a COSE Key from Section 7 of [RFC9052] and an implementation can choose to use a COSE Key representation if that is convenient.

4. Protocol Sub-Layers and Binding

The SAFE protocol operates with three distinct sub-layers, each with a different structure and purpose. They are described as follows, starting from the topmost sub-layer and working down toward the BP transport.

Activity Data: This sub-layer deals with the encoding of data items needed for each step of an activity and with how to progress an activity by taking received data from a peer from a previous step and generating data for a peer to be used in the next step. The activity data items are handled as a CBOR map using integer labels, and each activity type defines the meaning of labels in its data items.

Messaging: This sub-layer is used to identify each step of each

specific type of activity and provides the ability to acknowledge to a peer that a previous message was received and processed. Selective retransmission in SAFE occurs at the messaging sub-layer when a timer expires without receiving an acknowledgement.

Aggregation and Security: This sub-layer is related to aggregating SAFE messages into a single SAFE plaintext with optional padding, and then encrypting that into a single SAFE ciphertext. The use of padding allows an entity to ensure that all SAFE plaintext (and thus SAFE ciphertext) are the same size and avoid on-path traffic analysis.

Packetization: This sub-layer is used to embed either a single EDHOC message or single SAFE ciphertext along with identification metadata into a PDU for transport between SAFE entities.

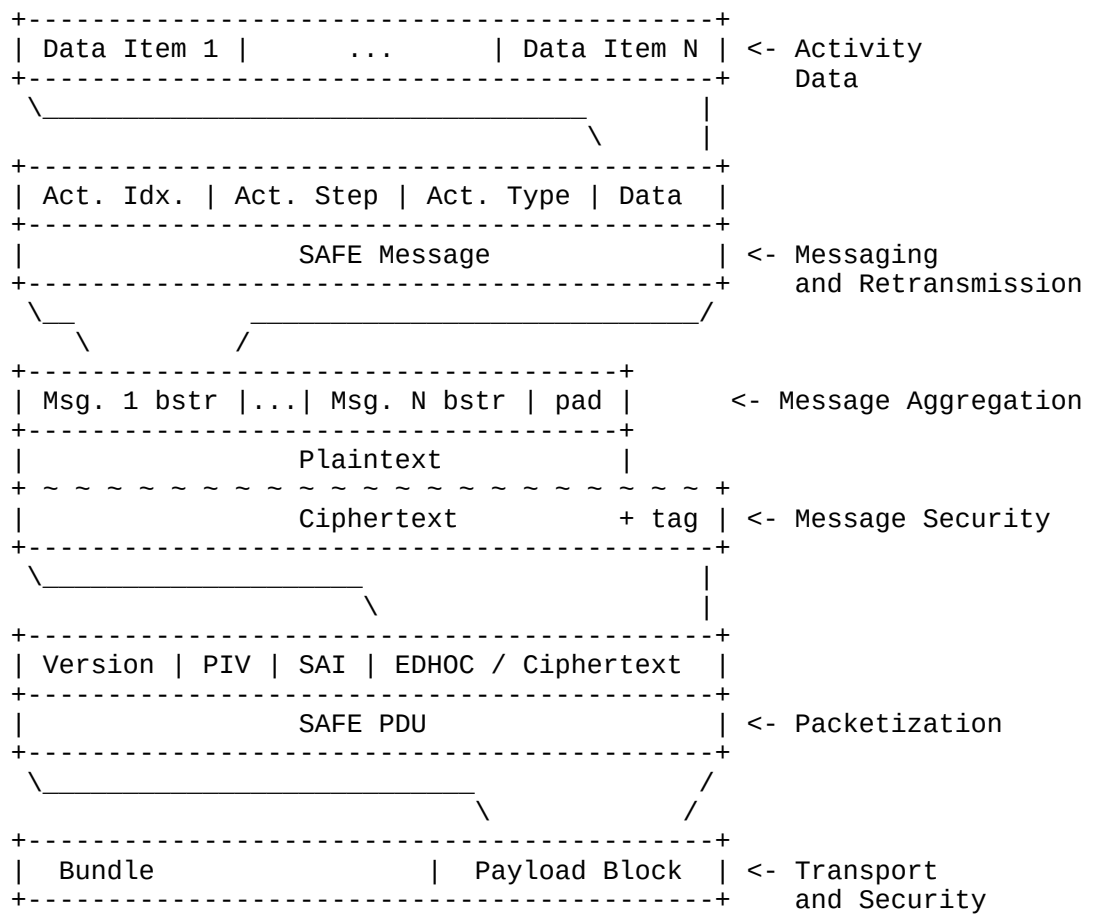


Figure 5: Breakdown of sub-layers within SAFE

4.1. Activity Sequencing

The contents of a SAFE message allow it to be correlated to a specific activity sequence and an individual step within that sequence (see Section 4.2 for details). All steps are numbered starting with zero at the initiator of an activity. The sending of a step number greater than zero is also used to acknowledge receipt and processing of the message with its preceding step number. The final acknowledgement of an activity is sent without a corresponding data payload in order to indicate that it is the end of the sequence.

Because SAFE allows messages to be aggregated into an PDU, this also enables explicit pipelining of multiple activities over a sequence of PDUs. It is an implementation matter to determine when to aggregate messages but the patterns defined in Section 7 make use of aggregation.

An example of this pipelining is shown in Figure 6, which depicts a series of PDUs sent in opposite direction between two peers "A" and "B". A Initial Authentication (IA) is the first activity, followed by a Capability Indication (CI) initiated in the opposite direction, and finally a pair of SA Creation (SC) activities initiated opposite that.

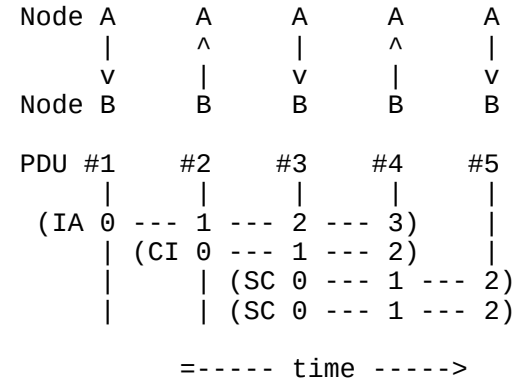


Figure 6: Pipelining of Activities

4.2. Message Structure

Each encoded SAFE message SHOULD use CBOR core deterministic encoding requirements from Section 4.2.1 of [RFC8949]. Each SAFE message SHALL consist of a CBOR sequence containing the following items.

Activity Index: This item is an unsigned integer used to correlate multiple messages associated with the same activity. Each activity SHALL be assigned a unique activity index when it is started by the initiator. A default algorithm to assign an activity index is to start at index zero for the first activity of a conversation and increment by one for each subsequent activity in that conversation.

Activity Step: This item is an unsigned integer used to distinguish messages for each step of an activity. The activity step value SHALL be limited to the inclusive range 0 to 65535. It is expected that most activity types will only involve two or three steps with a final acknowledgement, so the meaningful range of this value is much lower than the required range.

Activity Type: This item is a signed integer used to distinguish the purpose of the associated activity and of the data payload which follows. The activity type value SHALL be limited to the inclusive range -32768 to 32767. As defined in Section 12.3, positive values are for well-known activities, zero is reserved for the IA pseudo-activity (Section 5.1), and negative values are reserved for experimental or private use.

Data Payload: This item is either a CBOR map or a byte string, used to contain data specific to the logical step in the activity corresponding to the containing message.

The combination of Activity Index and Activity Step SHALL uniquely identify a message in an activity sequence independently of the type of activity being performed.

The activity index SHALL be unique per initiator and transport conversation (_e.g._, unordered pair of source and destination EID). The first activity index for a conversation SHALL be zero, and each subsequent activity initiated by a peer SHALL increment the activity index by one.

The first step of an activity SHALL be zero, and each subsequent step SHALL increment by one. This means that the initiator of an activity can be identified implicitly because it will only send messages with an even step number and will only receive messages with an odd step number.

The last message of an activity sequence SHALL NOT contain either an activity type or a data payload. That last message is purely to acknowledge the message from the previous step and conclude the activity.

This structure is indicated by the following CDDL for the general SAFE message structure and its data item payload.

```
safe-msg-bstr = bstr .cborseq safe-msg

safe-msg = $safe-msg .within safe-msg-struct
safe-msg-struct = [
    msg-ident,
    ? (
        act-type: int16,
        data-map
    )
]

; Unique identifier for a single message (one step of one activity)
msg-ident = (
    act-idx: uint,
    act-step: uint,
)

; activity-type-specific data
; non-negative labels are well-known
; negative labels are private use
data-map = {
    * label => value,
}
; Generic map label
label = int16
; Generic map value
value = any

; Signed integer that fits in 16-bit two’s complement form
int16 = (-32768 .. 32767) .within int
```

4.3. States, Deduplication, and Retransmission

As described in Section 3.2, the state kept by each entity about an activity includes the step of the message last sent by the entity (LTX) and the step last received from the peer entity (LRX). Based on these two step states, each entity can determine how to make progress

An example of this logic is shown in Figure 7. The state notation of {LTX,LRX} indicates the LTX and LRX steps respectively and negative values are used as a placeholder for an invalid/absent step number. If a transition has a specific trigger, it is indicated by square bracket text.

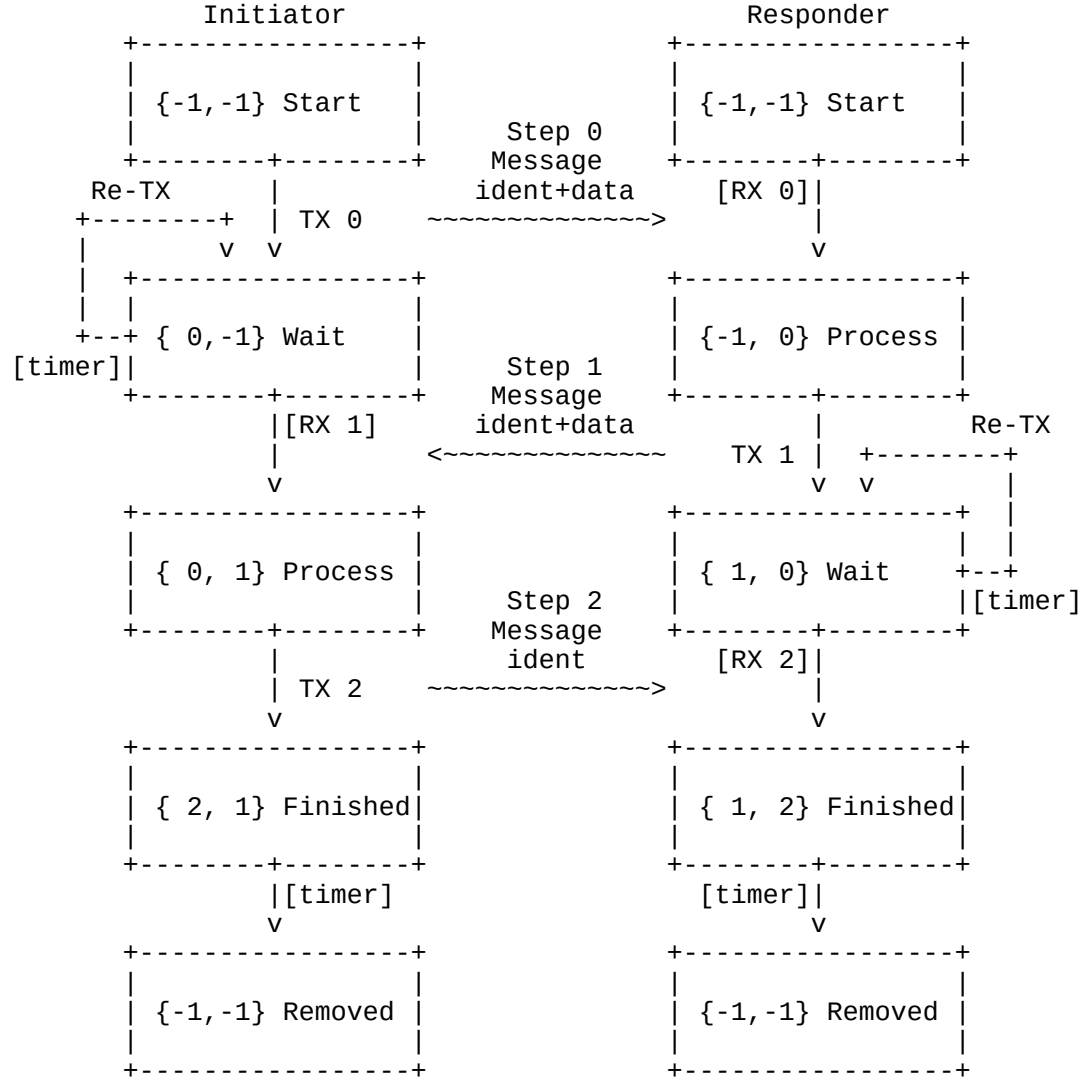


Figure 7: Example activity states for a two-step activity

Based on an existing activity state (Section 3.2) when the last LTX step is less than the LRX step, or when the initiator starts the activity, or an associated retransmission timer expires, the entity SHALL perform the following:

1. Compute the next step number as one more than the LRX step, if valid, or zero if the initiator has just started the activity.

2. If the next step is within the number of steps for this activity type, generate type-specific data items based on the activity type and next step.

   Otherwise, only the message identity is present and no activity type or data items are needed.

3. Encode and store an encoded SAFE message.

4. Send the SAFE message within a PDU (possibly combining with other messages to the same peer entity).

5. Update the LTX step to correspond with the sent message.

6. If either the LTX or LRX step are beyond the number of steps for this activity type, the activity is in the finished state.

   Otherwise, begin a retransmission timer correlated to the activity index with a duration taken from the expected round-trip time between the peer node plus some optional additional margin. It is an implementation matter to determine the specific margin added to the expected round-trip duration.

Upon receiving a SAFE message from a peer, an entity SHALL perform the following:

1. Decode at least the message identity (to be able to look up the activity state)

2. Determine which role this entity is performing by inspecting the step number: even steps are sent by the initiator and odd steps are sent by the responder

3. Use the peer EID and activity index to look up the specific activity state (Section 3.2) corresponding to the message

4. If the received message has a step less than or equal to the LRX of the activity state, it is ignored and this procedure is terminated

5. If present, decode and process the activity type and type-specific data items

6. Update the LRX step to correspond with the received message

7. If either the LTX or LRX step are beyond the number of steps for this activity type, the activity is in the finished state.

4.4. Message Aggregation and Security

Because SAFE is used to provide BPSec key material, the security properties of a SAFE bundle are more complex than other BP flows might be. For example, the bundles carrying Initialization messages need to be transported as plaintext payload (with intrinsic EDHOC protections) while other SAFE bundles need to be protected by the keys from a primary SA (negotiated as part of the EDHOC sequence).

The structure of a SAFE PDU is a CBOR sequence of the SAFE version number followed by header items and then payload items. The protocol in this document SHALL be identified by version number 1. The version specific header defined in this document SHALL be the following:

Partial IV: A partial initialization vector (IV) byte string or null value

Receiver SAI: A receiver security association identifier (SAI) byte string or integer (based on EDHOC compressed byte string encoding) or true value

When the Partial IV is null, the payload SHALL be one of the EDHOC messages defined in [RFC9528] and handled in accordance with Section 5.1. When the Receiver SAI is true, the payload SHALL be a Message 1 as defined in Section 5.2 of [RFC9528]. All other Receiver SAI values SHALL be treated as a connection identifier, encoded in accordance with Section 3.3.2 of [RFC9528], used to correlate with an existing EDHOC session.

When the Partial IV is a byte string, the payload SHALL be a SAFE ciphertext and handled in accordance with Section 9.1.2. In this case, Receiver SAI values SHALL be decoded as and treated as a Security Association Identifier (SAI) used to correlate with the Local SAI of a Primary SA (see Table 3) on the receiving entity.

This is indicated by the following CDDL for the SAFE bundle PDU sequence.

```
; Encoded to PDU as CBOR sequence
safe-pdu-seq = [
    version: 1,
    ; PDU variants follow the same structure with unique prefix items
    safe-pdu-edhoc // safe-pdu-confidential,
]

safe-pdu-edhoc //= (
    partial-iv: null,
    rx-sai: true,
    ; Group message_1 from RFC 9528
    message_1
)
safe-pdu-edhoc //= (
    partial-iv: null,
    rx-sai: safe-sai,
    edhoc_234 // error
)
; Equivalent to (message_2 // message_3 // message_4) from RFC 9528
; Encoded SAFE messages can be present in EAD items
edhoc_234 = bstr

safe-pdu-confidential //= (
    partial-iv: bstr,
    rx-sai: safe-sai,
    ; Ciphertext data corresponding to safe-pdu-plaintext below
    ciphertext: bstr
)

; A sequence of encoded-message bstr with optional tagged padding
safe-pdu-plaintext = (+ safe-msg-bstr, ? safe-padding)
safe-padding = #6.55799(bstr)

safe-pdu-aad = (
  rx-sai: safe-sai
)
```

Figure 8: SAFE PDU structure CDDL

Within restrictions defined for each message type, multiple messages MAY be combined into a single PDU (as either EDHOC EAD or SAFE plaintext). Due to the logic of the Initial Authentication (IA) sequencing, only one EDHOC session can make progress between two endpoints at any time so there is no concept of an EDHOC message embedded within an EAD item.

4.5. PDU Transport

Each SAFE PDU is handled an application data unit (ADU) of a BPv7 bundle, referred to as a "SAFE bundle" in this document. Additional constraints, controllability, and visibility on transport parameters are defined in the following subsection.

Both the source and destination EID, defined in Section 4.3.1 of [RFC9171], for a SAFE bundle SHALL be singleton. The source EID for SAND bundles SHALL NOT be a null EID. Each endpoint for SAND messaging needs to be an identified singleton. The bundle source and destination EID for received bundles SHALL be exposed to the SAND application in order to support message exchange sequencing.

When using the IPN scheme, the EIDs used as source and/or destination SHOULD use the well-known service number defined in Section 12.1. SAFE applications can use other schemes and service numbers, but such configuration is an implementation and deployment matter.

The bundle creation timestamp (both DTN time and sequence number), defined in Timestamp Section 4.3.1 of [RFC9171], for received bundles SHALL be exposed to the SAND application to allow it to order received SAND bundles.

5. Activity Types

This section defines the initial types of activity which make use of SAFE message (Section 4.2) sequencing to exchange data.

Any current or future activity type SHALL define how many steps comprise a single sequence of messages and what is the required data payload of each step.

5.1. Initial Authentication (IA)

This activity is used to establish an initial shared secret between two SAFE endpoints which don’t already have a usable SA, or when re-authentication is deemed necessary by either side of an existing SA.

Because this is the initializing activity for all other SAFE interactions, and occurs outside of any pre-existing SA, it does not follow the same message structure as other SAFE activities but it does use the same local progress bookkeeping and retransmission logic (from Section 4.3). The retransmission logic and the BP transport of Section 4 satisfies the EDHOC requirements of Section 3.4 of [RFC9528].

Even though the Initial Authentication (IA) activity does not follow the same messaging structure, it does have an activity identifier allocated to it in Table 22 to allow its state to be tracked in accordance with Section 3.2. The IA activity SHALL be identified by activity type 0.

Only one instance of this activity SHOULD be in progress at any time. The data of each PDU for the PI activity SHALL contain an EDHOC message as defined in Section 5 of [RFC9528].

The sequence of steps and their data for this activity are the following:

* Step 0: The PDU payload is an EDHOC Message 1 sequence.

* Step 1: The PDU payload is an EDHOC Message 2 byte string.

* Step 2: The PDU payload is an EDHOC Message 3 byte string.

* Step 3: The PDU payload is an EDHOC Message 4 byte string.

During transport, PI PDUs SHALL NOT be the target of a BPSec confidentiality operation between the participating (source and destination) nodes. The use of application-level confidentiality ensures the security of information exchanged via this PDU. There is no restriction on any intermediate security handling of SAFE PDUs between the participating nodes.

The CDDL corresponding to this activity type are the first two variations of safe-pdu-data from Section 4.4. Additionally, the _EAD_ payload of the EDHOC messages are extended to be able to carry encoded SAFE messages during the IA activity based on the safe-msg-bstr rule defined in Section 4.2.

5.1.1. EDHOC Requirements

After receiving the ID_CRED_x values from a peer, each SAFE entity SHALL

When the Message 2 payload is received by the initiator, both peers have established an EDHOC shared secret but only the responder has sent an identity to authenticate with the initiator. When the Message 3 payload is received by the responder, both peers have authenticated each other and established an application AEAD symmetric key and exporter state.

Upon the first reception or transmission of IA step 2 (_i.e._, EDHOC message_3), the entity SHALL create a primary SA based on the data derived in Section 8.1.

5.1.2. PKIX Profile

// TBD based on Section 4 of [I-D.ietf-dtn-bpsec-cose]

5.2. Capability Indication (CI)

This activity allows each entity to indicate its SAFE-related capabilities to its peer. The Capability Indication (CI) activity SHALL be identified by activity type 2.

The sequence of steps and their data for this activity are the following:

* Step 0: The payload is a map containing data items as defined below. Each item expresses a capability of the initiator entity.

* Step 1: The payload is a map containing data items as defined below. Each item expresses a capability of the responder entity.

* Step 2: No payload, message is acknowledgement.

Label	Type
1	A Concurrent Activity Support (CAS) limit
2	An EID Scheme Support (ESS) list
3	A BPSec Context Support (BCS) list

Table 5: CI Data Items

```
ci-data = { * $$ci-data } .within data-map

$$ci-data //= (1: concur-act-limit)
$$ci-data //= (2: eid-scheme-list)
$$ci-data //= (3: bpsec-ctxid-list)
```

5.3. SA Creation (SC)

This activity is used to create a new SA from within the context of an existing primary SA. The SA Creation (SC) activity SHALL be identified by activity type 3.

The sequence of steps and their data for this activity are the following:

* Step 0: The payload is a map containing data items as defined below. Some items (notably ESI, ESR, and KUS) MAY contain multiple proposals to allow the responder a selection.

* Step 1: The payload is a map containing data items as defined below. All items SHALL contain only a single proposal each.

* Step 2: No payload, message is acknowledgement.

The responder items with proposals SHALL be a logical subset of the initiator-provided proposals. The Secondary SA SHALL be configured to use a logical intersection between the items provided by the initiator and by the responder. For some cases of EID patterns, the intersection can be derived as a single pattern generated from the initiator and responder options. But for more complex cases, the intersection needs to be represented as a logical AND between the two separate patterns.

Label	Type
0	Error indications per Section 6.1 for messages after step 0
1	The local SAI per Section 6.5 for the SA which does not yet exist
Options for key material derivation	
2	An Additional Key Exchange (AKE) public key value
3	An Additional Random Nonce (ARN) value
Options for when to use the SA	
9	A Security Mode Selector (SMS) value
6	One or more Endpoint Selectors (ESx) proposals for the initiator side
7	One or more Endpoint Selectors (ESx) proposals for the responder side
8	A Node Time Interval (NTI) used as the SA validity time interval
Options for how to use the SA	
4	A Security Operation Selectors (SOS) with conditions for sourcing and accepting BPSec security
5	A Key Use Selectors (KUS) for a specific BPSec context, which contains one or more set of key use options as defined in Section 9

Table 6: SA Creation Data Items

```
sc-data = {* $$sc-data } .within data-map

; Error codes from the responder
$$sc-data //= (0: safe-ete)
; SAI for the sender
$$sc-data //= (1: safe-sai)

; Optional PRF parameters
$$sc-data //= (2: safe-ake)
$$sc-data //= (3: safe-arn)

; Initiator side endpoint selectors
$$sc-data //= (6: safe-esx)
; Responder side endpoint selectors
$$sc-data //= (7: safe-esx)
; Time limit for the SA
$$sc-data //= (8: safe-nti)

; BPSec Context and its options for how to use the keys
$$sc-data //= (4: safe-kus)
```

Upon the first reception or transmission of SC step 1, the entity SHALL create a secondary SA based on the data derived in Section 8.3.

5.4. SA Rekey (SR)

This activity is used to establish a new set of derived key (and possibly other) material within an existing SA without changing the other parameters of the SA (_e.g._, algorithm choice or endpoint selectors). The SA Rekey (SR) activity SHALL be identified by activity type 4.

The sequence of steps and their data for this activity are the following:

* Step 0: The payload is a map containing data items as defined below.

* Step 1: The payload is a map containing data items as defined below.

* Step 2: No payload, message is acknowledgement.

Label	Type
1	The local SAI per Section 6.5
2	A key exchange public key per Section 6.6
3	A random nonce per Section 6.7

Table 7: SA Rekey Data Items

```
sr-data = {* $$sr-data } .within data-map

; Error codes from the responder
$$sr-data //= (0: safe-ete)
; SAI for the sender
$$sr-data //= (1: safe-sai)

; Optional PRF parameters
$$sr-data //= (2: safe-ake)
$$sr-data //= (3: safe-arn)
```

Upon the first reception or transmission of SR step 1, the entity SHALL correlate the Local SAI of a Primary SA or Secondary SA and rekey it based on the data derived in Section 8.2 or Section 8.4 respectively.

5.5. SA Teardown (ST)

This activity is used to negotiate removal of an established SA from both entities. The SA Teardown (ST) activity SHALL be identified by activity type 5.

The sequence of steps and their data for this activity are the following:

* Step 0: The payload is a map containing data items as defined below.

* Step 1: The payload is a map containing data items as defined below.

* Step 2: No payload, message is acknowledgement.

Label	Type
1	The local SAI per Section 6.5

Table 8: SA Teardown Data Items

```
st-data = {* $$st-data } .within data-map

; Error codes from the responder
$$st-data //= (0: safe-ete)
; SAI for the sender
$$st-data //= (1: safe-sai / true)
```

5.6. Event Notification (EN)

// TBD This activity is used by the initiator to inform the peer entity of an event. The Event Notification (EN) activity SHALL be identified by activity type 7.

```
en-data = {* $$en-data } .within data-map

$$en-data //= (1: en-cause)
en-cause = [msg-ident]
```

6. Data Item Types

This section defines the initial set of data items which can be present as the data payload of a SAFE message (Section 4.2).

Some types of data item allow the activity initiator to provide multiple independent proposals, from which the responder can either choose one proposal or create a new single proposal which narrows down from one provided by the initiator. It is the responsibility for each data item type to define detailed logic of acceptable responder proposals. Each proposal SHALL take the form of a CBOR map. When multiple proposals are present, they SHALL take the form of a CBOR array of proposal maps.

```
safe-proposals = [data-map] / [+ data-map]
```

6.1. Error Type Enumeration (ETE)

The Error Type Enumeration (ETE) data item is used by its sender to signal a failure to process a message or make progress in an activity. The ETE value SHALL be an integer limited to the inclusive range -32768 to 32767.

These values are used in many activities to indicate a failure in decoding, processing, or consistency of received message contents. Well-known error values are non-negative and registered in Section 12.3. Private and experimental use error values are negative and not registered.

```
safe-ete = [+ safe-error]
safe-error = int16
```

6.2. Concurrent Activity Support (CAS)

This data item indicates how many concurrent (pipelined) SAFE activities the sender of the item supports. The minimum number of concurrent activities supported SHALL be 2. This is necessary in order to enable the processing of Capability Indication (CI) containing this data item during the Initial Authentication (IA) messaging. The maximum number of concurrent activities supported SHALL be 1024. This is an arbitrary upper bound not expected to be encountered during normal operations.

An attempt by a peer to send messages associated with more than this limit

```
concur-act-limit = 2 .. 1024
```

6.3. EID Scheme Support (ESS)

This data item indicates which EID schemes the sender of the item supports. Schemes are identified by the code points from the "Bundle Protocol URI Scheme Types" registry at [IANA-BUNDLE], which includes a block reserved for private-use. Expressing support for an EID scheme indicates that the node can handle EIDs of that scheme and EID Patterns with scheme-specific parts.

```
eid-scheme-list = [+ eid-scheme]
; Unrestricted per RFC 9171
eid-scheme = uint
```

6.4. BPSec Context Support (BCS)

This data item indicates which BPSec contexts the sender of the data item supports. Schemes are identified by the code points from the "BPSec Security Context Identifiers" registry at [IANA-BUNDLE], which includes a block of negative values reserved for private-use. Expressing support for a security context indicates that the node can handle sourcing, verifying, and accepting security blocks using that context.

```
bpsec-ctxid-list = [+ bpsec-ctxid]
; Restricted domain per RFC 9172
bpsec-ctxid = -32768 .. 32767
```

6.5. Security Association Identifier (SAI)

The Security Association Identifier (SAI) data item is used by both sides of an SA to uniquely identify the SA within each entity. This means that a single SA has two SAIs used to identify it, one from each entity.

An SAI SHALL be treated as an opaque byte string as its internal representation and comparison logic. An SAI SHALL have the same compressed encoding rules as EDHOC connection identifiers as defined in Section 3.3.2 of [RFC9528].

```
safe-sai = bstr / -24..23
```

6.6. Additional Key Exchange (AKE)

This data item indicates the public key of the sender for an additional ECDH exchange to add forward secrecy to a secondary SA. When sent by the activity initiator this is a request for additional key exchange, and when sent by the responder this is the confirmation that a key exchange is supported and desired. The algorithm and parameters related to key exchange are taken from the cipher suite negotiated by the primary SA.

```
; The compressed public key for the same algorithm as the primary SA
safe-ake = bstr
```

6.7. Additional Random Nonce (ARN)

This data item provides a random nonce value from the sender to add entropy into the PRF used during SA Creation (Section 5.3) and SA Rekey (Section 5.4) to generate SA data. The data value SHALL be a byte string with a size in the inclusive range 1 to 256 bytes.

```
safe-arn = bstr .size (1..256)
```

6.8. Security Mode Selector (SMS)

This data item controls how the secondary SA is used in the BP data plane by the BPSec entity in each participating node. This data value SHALL be an integer limited to the inclusive range -32768 to 32767.

Based on the discussion in Section 1.2 this document defines two modes: end-to-end (1) and one-hop (2). The behavior of these modes is defined in Section 9.2. Future specifications can add additional mode code points with different behavior.

```
safe-sms = secondary-sa-mode .within int16
secondary-sa-mode = &(
    end-to-end: 1,
    one-hop: 2,
)
```

6.9. Node Time Interval (NTI)

This data item filters on the current DTN time of the node hosting the SAFE entity. The type is used to limit the validity time of the associated SA, but requires synchronization of time between the two peers to perform properly. The start time can be the current time to allow immediate use of the SA or can be in the future to pre-establish an SA meant to be used later on. An entity can use NTI in multiple SCs, possibly using concurrent SC activities, to establish a chain of SAs each spanning a small time interval that together spans a large time interval.

The NTI value SHALL consist of a start time and an end time. The SA established with an NTI value SHALL be valid at an after the start time (inclusive), and only before the end time (exclusive).

```
// relocate this to secondary SA info section? Multiple SAs between
the same two peers with the same mode and endpoint selectors MAY have
validity NTIs which overlap in time. In that case, it is an
implementation matter to choose which SA to use for securing traffic.
// prefer lexicographic lower SAI?

; Using DTN time as reported by the local BP node
safe-nti = [
    begin: dtn-time,
    end: dtn-time
]
```

6.10. Endpoint Selectors (ESx)

This data item is used to determine which individual bundles will correlate with a secondary SA by comparing their source and destination EIDs to patterns. There is a separate endpoint selector applied to the initiator and responder side of a secondary SA, where each side selects on the source EID for traffic being forwarded from that node and destination EID for traffic being received by that node.

This selector filters on the Source or Destination of the bundle contained in its primary block (see Section 4.3.1 of [RFC9171]). The pattern can, but doesn't need to, include the EIDs from the node to which it applies (see Section 1.2).

The value of this data item consists of one or more EID Pattern in accordance with [I-D.ietf-dtn-eid-pattern], as indicated by the following CDDL. When multiple patterns are present they each represent an alternate proposal provided by the initiator and chosen by the responder of an activity.

```
safe-esx = eid-selectors .within safe-proposals
eid-selectors = embed-eid-pattern / [+ embed-eid-pattern]
```

6.11. Security Operation Selectors (SOS)

This data item controls how the node of each SAFE endpoint applies BPSec security operations to traffic selected for handling by the secondary SA.

The SOS value SHALL be an array with the following items.

1. A list of target block types to which the security will be applied. Block type zero SHALL be used to refer to the primary block.

2. The type of security service being sourced by the forwarding entity and accepted by the receiving entity. This value is either integrity (1) or confidentiality (2), as defined in [RFC9172].

```
safe-sos = [
    target-block-types: [+ bpv7-block-type],
    service: bpsec-service,
]
; Type consistent with RFC 9171
bpv7-block-type = uint
; Services available in RFC 9172
bpsec-service = &(
    integrity: 1,
    confidentiality: 2,
)
```

6.12. Key Use Selectors (KUS)

This data item is used to negotiate the purpose(s) for which an SA-derived symmetric key can be used by each side of the SA.

The KUS value is an array with the following items.

1. A single BPSec Context Identifier value in accordance with the registry in [IANA-BUNDLE].

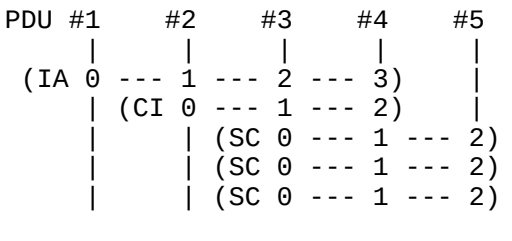
2. One or more maps of context-specific options, each representing an alternate proposal provided by the initiator and chosen by the responder of an activity. The interpretation of these options is specific to each BPSec Context ID. The initial set of supported contexts is defined in Section 9.

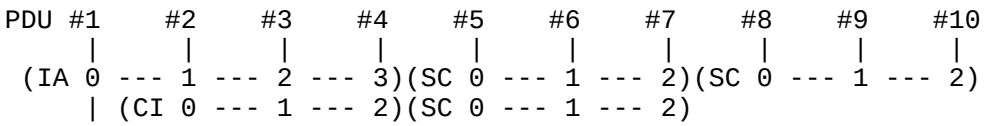
```
safe-kus = $safe-kus .within safe-kus-struct
safe-kus-struct = [
    ctxid: bpsec-ctxid,
    options: data-map / [+ data-map] .within safe-proposals
]
```

Some examples of what KUS options can be defined by each Context binding are: a specific algorithm identifier or an choice of additional authenticated data (AAD) outside the security target block.

7. Activity Patterns

The packetization form defined in Section 4 allows multiple messages to be combined into a single ADU but this behavior is limited by support for the receiving entity, as defined and communicated in the Concurrent Activity Support (CAS) data item. Two examples of how this parameter affects activity sequencing is shown in Figure 9 for a large CAS limit and Figure 10 for a constraining limit of two.





Regardless of how activities are aggregated into specific ADUs, the following requirements apply to ordering of activities related to a single primary SA.

When no primary SA exists between two SAFE application endpoints, the first messaging between those applications SHALL be Initial Authentication (IA) used to progress the activity state and establish a primary SA. After a primary SA exists, either side of the SA MAY choose to initialize a new primary SA and re-authenticate with its peer.

Only after the first step of the IA activity has been received, the responder for the IA activity SHALL begin a Capability Indication (CI) activity. This CI activity MAY begin immediately upon sending of the second IA step, or MAY wait until after the IA activity has completed and a primary SA is established.

Only after the first step of the CI activity has been received, the initiator for the IA activity SHOULD begin any number of needed SA Creation (SC) activities. Only after the second step of the CI activity has been received, the responder for the IA activity SHOULD begin any number of needed SA Creation (SC) activities. The number of concurrent SC activities allowed to be started at each step is limited by the CAS limit for the receiving peer.

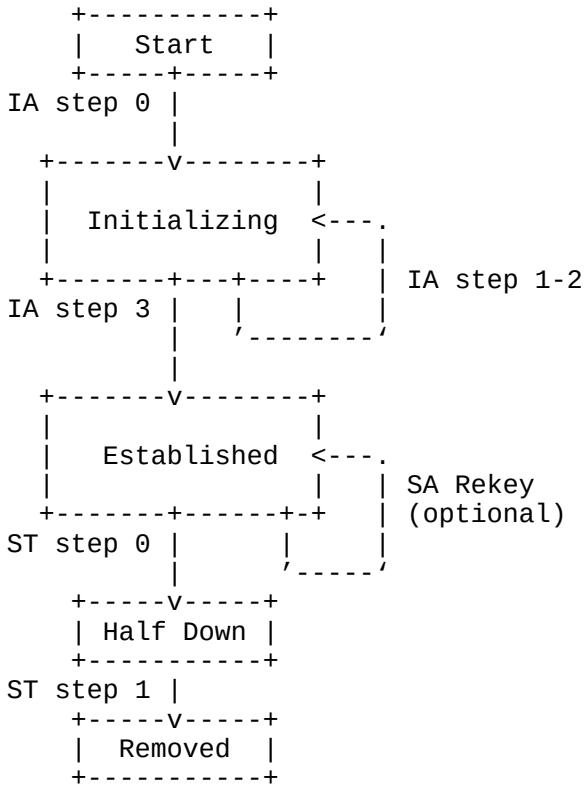


Figure 11: Notional States of a Primary SA

8. Shared Secret Derivations

Within the Initial Authentication (IA) activity, a shared secret is established and internal pseudo-random keys (PRKs) for EDHOC processing are derived. One of the final PRKs, the PRK_exporter, is used for "EDHOC application" purposes which in this case means SAFE entity use.

8.1. Primary SA Creation

The primary SA is used to manage derived data for two independent purposes: key parameters for message security between the SAFE endpoints themselves, and as the PRK shared secret used to derive secondary SAs between the two SAFE entities.

The primary SA key uses take the form of COSE key objects as defined in Section 7 of [RFC9052]. Because the acceptable algorithms are limited to AEAD encryption, the acceptable operations for each key in each entity are one of: encrypt (3), or decrypt (4).

When the COSE algorithm is one of the AEAD algorithms, the COSE key SHALL contain the following parameters. Notably, these keys do not contain a kid parameter because they are not intended to be referenced from outside of the SAFE entity managing the primary SA.

kty (1):
The value Symmetric (4).

alg (3):
The negotiated "application AEAD algorithm" from the IA activity.

key_ops (4):
Either encrypt (3) if this entity is the IA initiator and the key is for traffic from the initiator side, or decrypt (4) otherwise

Base IV (5):
Either BIV_IR (from Figure 12) if the key is for traffic from the initiator side, or BIV_RI otherwise

k (-1):
Either K_IR (from Figure 12) if the key is for traffic from the initiator side, or K_RI otherwise

Pseudocode used to explain this is shown in Figure 12, where the named parameters are:

key_length: The length of the encryption key for the "application AEAD algorithm" of the EDHOC cipher suite.

iv_length: The length of the initialization vector for the "application AEAD algorithm" of the EDHOC cipher suite.

hash_length: The length of output size of the "application hash algorithm" of the EDHOC cipher suite.

```
K_IR   = EDHOC_Exporter(TBA4, 'key_ir', key_length)
BIV_IR = EDHOC_Exporter(TBA4, 'biv_ir', iv_length)

K_RI   = EDHOC_Exporter(TBA4, 'key_ri', key_length)
BIV_RI = EDHOC_Exporter(TBA4, 'biv_ri', iv_length)

PRK_SA1 = EDHOC_Exporter(TBA4, 'prk_sa1', hash_length)
```

Figure 12: Primary SA Creation

The "_ir" derived values apply to SAFE traffic from the initiator to the responder of the IA activity, and the "_ri" derived values apply to SAFE traffic in the other direction.

8.2. Primary SA Rekey

This procedure is based on the example from Appendix H of [RFC9528].

```
context_rekey = ARN(i) | ARN(r) | G_XY
PRK_out, PRK_exporter = EDHOC_KeyUpdate(context_rekey)
```

Figure 13: Primary SA Rekey

8.3. Secondary SA Creation

An application-level derivation function SAFE_KDF is defined with the same logic as the EDHOC_KDF in Section 4.1.2 of [RFC9528] except using the "application hash algorithm" of the EDHOC cipher suite. This KDF is used once per secondary SA to derive a seed PRK for the entire SA.

Pseudocode used to explain this is shown in Figure 14, where the named parameters are:

SAI(i) and SAI(r): The Security Association Identifier (SAI) from the initiator and responder respectively

ARN(i) and ARN(r): The Additional Random Nonce (ARN) from the initiator and responder respectively, or the empty byte string if no such data item was received

G_XY: The shared secret derived from Additional Key Exchange (AKE) from both the initiator and responder, or the empty byte string if AKE was not received from both entities

```
context_sa2 = SAI(i) | SAI(r) | ARN(i) | ARN(r) | G_XY
PRK_SA2 = SAFE_KDF(PRK_SA1, 0, context_sa2, hash_length)
```

Figure 14: Secondary SA derived PRK

Additional output key material (OKM) specific to the BPSec security context being used is then derived from the PRK_SA2 value with integer labels for different purposes. It is part of the obligation of each BPSec key use to define what label values apply to key material needed by that context. An IANA registry for recording these labels is defined in Section 12.3.

Pseudocode used to explain this is shown in Figure 15, where the named parameters are:

ctxid: The BPSec Context Identifier as an unsigned integer

context: The BPSec-context-specific context byte string, which can be the empty byte string if not needed

length: The length of output key material being derived

```
SAFE_OKM(ctxid, context, length)
  = SAFE_KDF(PRK_SA2, ctxid, context, length)
```

Figure 15: Secondary SA derived OKM

8.4. Secondary SA Rekey

// TBD

9. Security Association Uses

The ultimate point of establishing a SA is to make use of the derived symmetric keys for BPSec security operations as defined in [RFC9172].

When BPSec contexts make use of SAs defined by this document they will require an explicit mapping from the algorithm and operation selectors from Section 3.4 onto the algorithm and operation identifiers specific to that context. For example the default security contexts correspond to a subset of the AES-GCM algorithms of specific key lengths (see Section 9.4) and the HMAC-SHA2 algorithms of specific key lengths (see Section 9.3).

All current and future BPSec context bindings SHALL define the following aspects:

* State to be managed by a SAFE entity under Key Use Options and Key Material fields of the Secondary Security Associations and how that key material is derived from the PRK of the secondary SA.

* Data items to be exchanged within the SA Creation (SC) messages to negotiate the Key Use Selectors.

* Detailed requirements about how the Key Material is derived from the SAFE_SA2 PRK from the secondary SA.

* Detailed requirements about how the secondary SA key information is used by the context in all BPSec roles: security source, verifier, and acceptor.

9.1. SAFE Confidentiality

Part of the purpose of establishing a primary SA with a peer is to derive an encryption key (with associated base IV) to enable SAFE message confidentiality in each direction. Similar to how EDHOC cryptographic functions make use of COSE primitives but not COSE message encodings, SAFE reuses COSE primitives for its own end-to-end security.

9.1.1. Without a Primary SA

These conditions apply even if there is an existing Primary SA which is being superseded by a new IA activity.

While an IA activity is in progress, outgoing SAFE messages SHALL be embedded as EAD items within an associated EDHOC message. Each encoded message byte string SHALL be placed in a separate EAD value with corresponding EAD label TBA5 in accordance with Section 3.8 of [RFC9528].

While an IA activity is in progress, incoming SAFE messages SHALL be extracted from the EAD of an associated EDHOC message. These messages are not processed any differently than other received SAFE messages.

9.1.2. With a Primary SA

After the IA activity has completed (initiator has sent, responder has received) step 3, a SAFE entity SHALL encrypt aggregated outgoing messages according to the following.

1. Increment the Partial IV counter associated with the peer entity.

2. Encode the Partial IV as a minimum-length byte string in network byte order.

3. Iterate through and encode a CBOR sequence of byte strings items for all messages in the plaintext.

4. Internally generate and encrypt a COSE_Encrypt0 object as defined in Section 5.3 of [RFC9052] using the following inputs:

   Protected Header:
      An empty byte string h''

   Unprotected Header:
      A map containing the following pairs:

      alg (1):
         The "application AEAD algorithm" of the EDHOC cipher suite from the Primary SA

      Partial IV (6):
         The Partial IV value computed in the previous step

   key:
      Either K_IR or K_RI of Section 8.1 from the Primary SA for the initiator or responder respectively, which will include a Base IV key parameter.

   plaintext:
      The encoded plaintext from above

   external_aad:
      A byte string encoding the CBOR Sequence safe-pdu-aad (as defined in Figure 8, and using the same encoding as the PDU)
      // No binding to transport source EID??

5. Use the result to construct the following PDU fields, named as in Section 4.4:

   partial-iv: The Partial IV unprotected header byte string encoded in accordance with Section 3.3.2 of [RFC9528]

   rx-sai: The Peer SAI field of the Primary SA

   ciphertext: The COSE-encrypted ciphertext from the previous step

Because this encoding does not use AAD to bind it to specific transport parameters, the entire PDU can be retained and retransmitted if necessary (see Section 4.3).

Upon receiving an PDU containing a non-null partial-iv value, the SAFE entity SHALL decrypt the aggregated messages according to the following.

1. Use the provided rx-sai to match with the Local SAI of one Primary SA.

   If no match is found then
   // TBD.

2. Internally generate and decrypt a COSE_Encrypt0 object as defined in Section 5.3 of [RFC9052] using the following parameters:

   Protected Header:
      An empty byte string h''

   Unprotected Header:
      A map containing the following pairs:

      alg (1):
         The "application AEAD algorithm" of the EDHOC cipher suite from the Primary SA

      Partial IV (6):
         The partial-iv value from the SAFE PDU header

   key:
      Either K_IR or K_RI of Section 8.3 from the Secondary SA for the initiator or responder respectively.

   ciphertext:
      The ciphertext field of the PDU payload

   external_aad:
      A byte string encoding the CBOR Sequence safe-pdu-aad (as defined in Figure 8, and using the same encoding as the PDU)

   If decryption fails then
   // TBD.

3. Iterate through, decode, and process the CBOR sequence of byte strings items for all messages in the plaintext.

For both encryption and decryption, the Base IV (defined in Section 8.1) present in the keys of the Primary SA is combined with the Partial IV byte string in accordance with Section 3.1 of [RFC9052].

9.2. Common BPSec Operation

This section contains logic related to how the Security Mode Selector (SMS), Validity Time Interval, Endpoint Selectors, and Security Operation Selector (SOS) of the Secondary SA (as described in Section 3.4) inform the BPSec entity on both the Initiator and Responder nodes.

The SMS value determines at which of the BP Agent interaction points (see Figure 2) the SA is applicable with logic as follows.

1. When the SMS value is end-to-end (1), the node SHALL apply the SA policy at bundle transmission (from endpoint application) as source and at bundle delivery (to endpoint application) as acceptor.

   When the SMS value is one-hop (2), the node SHALL apply the SA policy at bundle forwarding (to CLA) as source and at bundle reception (from CLA) as acceptor.

2. At the source interaction point, the node SHALL only apply the SA policy when the current DTN time of the node is within the Validity Time Interval of the SA.

   At the acceptor interaction point, the node SHALL only apply the SA policy when the DTN time from the Timestamp of the Primary Block (see Section 4.3.1 of [RFC9171]) is within the Validity Time Interval of the SA.

3. At each interaction point, the node SHALL compare the Source and Destination EID of the Primary Block (as defined in Section 4.3.1 of [RFC9171]) of each bundle against one of the following.

   At the source interaction point, the Source EID is compared to the Local Endpoint Selectors field and the Destination EID is compared to the Peer Endpoint Selectors field.

   At the acceptor interaction point, the Source EID is compared to the Peer Endpoint Selectors field and the Destination EID is compared to the Local Endpoint Selectors field.

4. If the bundle has passed the above checks, the Security Operation Selector is used to determine which blocks to apply the SA as security source or acceptor. If any of the target block types is not present in the bundle,
   // TBD failure?

5. For each of the target block numbers filtered-in by the previous step, the node SHALL apply the service identified by the SOS of the SA, using the Context ID, Key Use Options, and Key Information as defined for each context below.

9.3. Binding for BIB-HMAC-SHA2 Context

This section defines how a secondary SA can be negotiated for and used by the integrity context of [RFC9173] with context identifier code point 1.

9.3.1. Secondary SA Information

For the BIB-HMAC-SHA2 context the "Key Use Options" field of Section 3.4 is augmented to include the following information.

Name	Description
SHA Variant	One of the code points defined in Section 3.3.1 of [RFC9173].
Integrity Scope Flags	A value with bit flags defined in Section 3.3.3 of [RFC9173].

Table 9

For the BIB-HMAC-SHA2 context the "Key Information" field of Section 3.4 is augmented to include the following information.

Name	Description
TX Key	A byte string for the security source role.
RX Keys	One or more byte strings for the security acceptor role.

Table 10

9.3.2. KUS Options

The BIB-HMAC-SHA2 context uses existing code points to identify key use options.

```
$safe-kus /= [
    ctxid: 1,
    options: kus-options-ctxid1 .within safe-proposals
]
kus-options-ctxid1 = kus-map-ctxid1 / [+ kus-map-ctxid1]
kus-map-ctxid1 = {
    ; SHA Variant values from Section 3.3.1 of RFC 9173
    1: uint,
    ; Integrity Scope flags from Section 3.3.3 of RFC 9173
    2: uint,
}
```

9.3.3. Key Derivation

The secondary SA key uses for context 1 take the form of raw symmetric key data. Because the acceptable algorithms are limited to HMAC, there are no other options beyond the symmetric key.

The raw symmetric key SHALL be either CTX1_KEY_IR (from Figure 16) if the key is for traffic from the initiator side, or CTX1_KEY_RI otherwise.

```
CTX1_KEY_IR = SAFE_OKM(1, 'key_ir', key_length)
CTX1_KEY_RI = SAFE_OKM(1, 'key_ri', key_length)
```

Figure 16: Secondary SA BIB-HMAC-SHA2 Context Derivations

9.3.4. BPSec Operation

When each security operation for context 1 needs to be applied, as defined in Section 9.2, as the security source the node SHALL perform the following:

1. A BIB parameter for SHA Variant (1) SHALL be constructed from the SHA Variant option of the SA.

2. A BIB parameter for Integrity Scope Flags (3) SHALL be constructed from the Integrity Scope option of the SA.

3. A BIB result for Expected HMAC (1) SHALL be constructed in accordance with the structure of Section 3.4 of [RFC9173] and content of Section 3.8.1 of [RFC9173].

When each security operation for context 1 needs to be applied, as defined in Section 9.2, as the security acceptor the node SHALL perform the following:

1. A BIB parameter for SHA Variant (1) SHALL be extracted and compared to the SHA Variant option of the SA. If the received SHA Variant differs from the SA, the procedure is considered failed.

2. A BIB parameter for Integrity Scope Flags (3) SHALL be extracted and compared to the Integrity Scope option of the SA. If the received Integrity Scope Flags differs from the SA, the procedure is considered failed.

3. A BIB result for Expected HMAC (1) SHALL be extracted and verified in accordance with Section 3.8.2 of [RFC9173]. If the verification fails, the procedure is considered failed.

9.4. Binding for BCB-AES-GCM Context

This section defines how a secondary SA can be negotiated for and used by the confidentiality context of [RFC9173] with context identifier code point 2.

9.4.1. Secondary SA Information

For the BCB-AES-GCM context the "Key Use Options" field of Section 3.4 is augmented to include the following information.

Name	Description
AES Variant	One of the code points defined in Section 4.3.2 of [RFC9173].
AAD Scope Flags	A value with bit flags defined in Section 4.3.4 of [RFC9173].
IV Counter	An integer counter which initializes to 0 at SA creation.

Table 11

For the BCB-AES-GCM context the "Key Information" field of Section 3.4 is augmented to include the following information.

Name	Description
TX Key	A byte string for the security source role.

TX IV Counter	An integer counter which initializes to 0 at SA creation.
RX Keys	One or more byte strings for the security acceptor role.

Table 12

9.4.2. KUS Options

The BCB-AES-GCM context uses existing code points to identify key use options.

```
$safe-kus /= [
    ctxid: 2,
    options: kus-options-ctxid2 .within safe-proposals
]
kus-options-ctxid2 = kus-map-ctxid2 / [+ kus-map-ctxid2]
kus-map-ctxid2 = {
    ; AES Variant values from Section 4.3.2 of RFC 9173
    1: uint,
    ; AAD Scope flags from Section 4.3.4 of RFC 9173
    2: uint,
}
```

9.4.3. Key Derivation

The secondary SA key uses for context 2 take the form of raw symmetric key data. Because the acceptable algorithms are limited to AEAD, there are no other options beyond the symmetric key.

The raw symmetric key SHALL be either CTX2_KEY_IR (from Figure 17) if the key is for traffic from the initiator side, or CTX2_KEY_RI otherwise.

```
CTX2_KEY_IR = SAFE_OKM(2, 'key_ir', key_length)
CTX2_KEY_RI = SAFE_OKM(2, 'key_ri', key_length)
```

Figure 17: Secondary SA BCB-AES-GCM Context Derivations

9.4.4. BPSec Operation

When each security operation for context 2 needs to be applied, as defined in Section 9.2, as the security source the node SHALL perform the following:

1. A BIB parameter for AES Variant (2) SHALL be constructed from the AES Variant option of the SA.

2. A BIB parameter for AAD Scope Flags (4) SHALL be constructed from the AAD Scope option of the SA.

3. Increment the TX IV Counter associated with the Key Information of the SA.

4. Encode the TX IV Counter as a fixed-length 12-byte string in network byte order.

5. A BIB parameter for Initialization Vector (1) SHALL be constructed from the encoded IV Counter of the previous step.

6. The encryption SHALL be performed, modifying the target block in place, in accordance with Section 4.8.1 of [RFC9173].

7. A BIB result for Authentication Tag (1) SHALL be constructed in accordance with the structure of Section 4.4.1 of [RFC9173] and content of Section 4.8.1 of [RFC9173].

When each security operation for context 2 needs to be applied, as defined in Section 9.2, as the security acceptor the node SHALL perform the following:

1. A BCB parameter for AES Variant (2) SHALL be extracted and compared to the AES Variant option of the SA. If the received AES Variant differs from the SA, the procedure is considered failed.

2. A BCB parameter for AAD Scope Flags (3) SHALL be extracted and compared to the AAD Scope option of the SA. If the received AAD Scope Flags value differs from the SA, the procedure is considered failed.

3. A BCB result for Authentication Tag (1) SHALL be extracted and the decryption procedure of Section 4.8.2 of [RFC9173] performed. If the decryption fails, the procedure is considered failed.

9.5. Binding for COSE Context

This section defines how a secondary SA can be negotiated for and used by the COSE context of [I-D.ietf-dtn-bpsec-cose] with context identifier code point 3. The derived keys are suitable for use with COSE encryption or MAC operations between the nodes hosting the SAFE entities.

9.5.1. Secondary SA Information

For the COSE context the "Key Use Options" field of Section 3.4 is augmented to include the following information.

Name	Description
COSE Algorithm	One of the code points defined in the "COSE Algorithms" registry of [IANA-COSE] for either AEAD encryption or MAC operation.
AAD Scope	A map structure defined in Section 2.2.2 of [I-D.ietf-dtn-bpsec-cose].

Table 13

For the COSE context the "Key Information" field of Section 3.4 is augmented to include the following information.

Name	Description
TX COSE Key	A COSE Key object for the security source role.
TX Partial IV Counter	An integer counter which initializes to 0 at SA creation. This field is used only for confidentiality services.
RX COSE Keys	One or more COSE Key objects for the security acceptor role.

Table 14

The COSE key structure is defined in Section 7 of [RFC9052].

9.5.2. KUS Options

The BPSec COSE context uses existing COSE code points to identify key use options.

Label	Description
3	One of the code points defined in the "COSE Algorithms" registry of [IANA-COSE] for either AEAD encryption or MAC operation.

Table 15: COSE Context KUS Options

```
$safe-kus /= [
    ctxid: 3,
    options: kus-options-ctxid3 .within safe-proposals
]
kus-options-ctxid3 = kus-map-ctxid3 / [+ kus-map-ctxid3]
kus-map-ctxid3 = {
    ; COSE alg header values
    3: tstr / int,
}
```

9.5.3. Key Derivation

The secondary SA key uses for context 3 take the form of COSE key objects. Because the acceptable algorithms are limited to AEAD encryption and MAC, the acceptable operations for each key in each entity are one of: encrypt (3), decrypt (4), MAC create (9), or MAC verify (10).

When the COSE algorithm is one of the MAC algorithms, the COSE key SHALL contain the following parameters.

kty (1):
: The value Symmetric (4).

kid (2):

: // TBD

alg (3):
: The negotiated algorithm from the KUS options.

key_ops (4):
: Either MAC create (9) if this entity is the SC initiator and the key is for traffic from the initiator side, or MAC verify (10) otherwise

k (-1):
Either CTX3_KEY_IR (from Figure 18) if the key is for traffic from the initiator side, or CTX3_KEY_RI otherwise

When the COSE algorithm is one of the AEAD algorithms, the COSE key SHALL contain the following parameters.

kty (1):
The value Symmetric (4).

kid (2):

// TBD

alg (3):
The negotiated algorithm from the KUS options.

key_ops (4):
Either encrypt (3) if this entity is the SC initiator and the key is for traffic from the initiator side, or decrypt (4) otherwise

Base IV (5):
Either CTX3_BIV_IR (from Figure 18) if the key is for traffic from the initiator side, or CTX3_BIV_RI otherwise

k (-1):
Either CTX3_KEY_IR (from Figure 18) if the key is for traffic from the initiator side, or CTX3_KEY_RI otherwise

```
CTX3_KEY_IR = SAFE_OKM(3, 'key_ir', key_length)
CTX3_BIV_IR = SAFE_OKM(3, 'biv_ir', iv_length)

CTX3_KEY_RI = SAFE_OKM(3, 'key_ri', key_length)
CTX3_BIV_RI = SAFE_OKM(3, 'biv_ri', iv_length)
```

Figure 18: Secondary SA COSE Context Derivations

9.5.4. BPSec Operation

When each security operation for context 3 needs to be applied, as defined in Section 9.2, as the security source and the COSE algorithm is one of the MAC algorithms, the node SHALL perform the following:

1. A BIB parameter for AAD Scope (5) SHALL be constructed from the AAD Scope option of the SA.

2. A BIB result for COSE_Mac0 (17) SHALL be constructed in accordance with Section 2.3.1 of [I-D.ietf-dtn-bpsec-cose] containing the following parameters.

   Protected Header:
      An encoded map containing the following pairs:

      alg (1):
         The COSE algorithm value from the SA

   Unprotected Header:
      A map containing the following pairs:

      kid (4):
         The kid parameter from the COSE Key of the TX Key Information of the SA

      kid context (10):
         The Peer SAI of the SA as a direct byte string

   key:
      The COSE Key of the TX Key Information of the SA.

   payload and external_aad:
      Taken from the target block and AAD Scope option in accordance with Section 2.3.1 of [I-D.ietf-dtn-bpsec-cose]. The actual payload will be detached and taken from the target block.

When each security operation for context 3 needs to be applied, as defined in Section 9.2, as the security acceptor and the COSE algorithm is one of the MAC algorithms, the node SHALL perform the following:

1. A BIB parameter for AAD Scope (5) SHALL be extracted and compared to the AAD Scope option of the SA. If the received AAD Scope value differs from the SA, the procedure is considered failed.

2. A BIB result for COSE_Mac0 (17) SHALL be extracted in accordance with Section 2.3.1 of [I-D.ietf-dtn-bpsec-cose] and verified to have the following minimum parameters.

   Protected Header:
      An encoded map containing the following pairs:

      alg (1):
         The COSE algorithm value matching that of the SA

   Unprotected Header:

A map containing the following pairs:

kid (4):
A value matching a kid parameter from one of the COSE Key of the TX Key Information of the SA

If the required header parameters are missing or invalid, the procedure is considered failed.

3. The extracted COSE_Mac0 is then verified using the following parameters:

key: The matching COSE Key from the TX Key Information of the SA.

payload and external_aad: Taken from the target block and AAD Scope option in accordance with Section 2.3.1 of [I-D.ietf-dtn-bpsec-cose]. The actual payload will be detached and taken from the target block.

If the verification fails, the procedure is considered failed.

When each security operation for context 3 needs to be applied, as defined in Section 9.2, as the security source and the COSE algorithm is one of the AEAD algorithms, the node SHALL perform the following:

1. A BCB parameter for AAD Scope SHALL be constructed from the AAD Scope option of the SA.

2. Increment the TX Partial IV Counter associated with the Key Information of the SA.

3. Encode the TX Partial IV Counter as a minimum-length byte string in network byte order.

4. A BCB result for COSE_Encrypt0 (16) SHALL be constructed in accordance with Section 2.3.2 of [I-D.ietf-dtn-bpsec-cose] containing the following parameters.

Protected Header:
An encoded map containing the following pairs:

alg (1):
The COSE algorithm value from the SA

Unprotected Header:
A map containing the following pairs:

kid (4):
The kid parameter from the COSE Key of the TX Key Information of the SA

partial iv (6):
The encoded Partial IV from the earlier step

key:
The COSE Key of the TX Key Information of the SA, which includes a Base IV parameter.

payload and external_aad:
Taken from the target block and AAD Scope option in accordance with Section 2.3.2 of [I-D.ietf-dtn-bpsec-cose]. The final payload will be detached.

5. The constructed COSE_Encrypt0 SHALL be encrypted, modifying the target block in place as a detached payload, in accordance with Section 2.3.2 of [I-D.ietf-dtn-bpsec-cose].

When each security operation for context 3 needs to be applied, as defined in Section 9.2, as the security acceptor and the COSE algorithm is one of the AEAD algorithms, the node SHALL perform the following:

1. A BCB parameter for AAD Scope (5) SHALL be extracted and compared to the AAD Scope option of the SA. If the received AAD Scope value differs from the SA, the procedure is considered failed.

2. A BCB result for COSE_Encrypt0 (16) SHALL be extracted in accordance with Section 2.3.2 of [I-D.ietf-dtn-bpsec-cose] and verified to have the following minimum parameters.

Protected Header:
An encoded map containing the following pairs:

alg (1):
The COSE algorithm value matching that of the SA

Unprotected Header:
A map containing the following pairs:

kid (4):
A value matching a kid parameter from one of the COSE Key of the RX Key Information of the SA

partial iv (6):
A byte string Partial IV value

If the required header parameters are missing or invalid, the procedure is considered failed.

3. The extracted COSE_Encrypt0 is then processed to decrypt using the following parameters:

   key: The COSE Key of the RX Key Information of the SA, which includes a Base IV parameter.

   payload and external_aad: Taken from the target block and AAD Scope option in accordance with Section 2.3.2 of [I-D.ietf-dtn-bpsec-cose]. The final payload will be detached.

   The decryption modifies the target block in place as a detached payload. If the authenticated decryption fails, the procedure is considered failed.

The parameters of the derived COSE keys already intrinsically restrict their use based on the requirements of [I-D.ietf-dtn-bpsec-cose] and [RFC9053], so no additional checks are specified in this document.

10. PKIX Certificate Profile

// TBD with a new extended key use for this protocol. Profile of [RFC5280] allowing C509 [I-D.ietf-cose-cbor-encoded-cert].

11. Security Considerations

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

11.1. Threat: Passive Leak of Data

Because the SAFE protocol uses EDHOC for its initial authentication activity and messaging and uses primary SA data for confidentiality afterward, the only information which is exposed in plaintext are the Security Association Identifier (SAI) values (which are also EDHOC connection identifiers) and SAFE confidentiality partial IV (Section 4.4) values.

Both of these values are used strictly for uniqueness and can either be generated by a SAFE entity deterministically or randomly. Neither value contains data derived from or correlated to any outside information, so visibility to a passive attacker is not useful to degrade SAFE security. Their visibility could be used by a passive attacker for SAFE traffic analysis.

The sizes of ciphertext values in all forms of SAFE PDU can be used by a passive attacker to estimate the types of SAFE activities being performed. To mitigate this threat, it is possible for a SAFE entity to use EDHOC padding EAD (see Section 3.8.1 of [RFC9528]) during the IA activity or SAFE confidential PDU plaintext padding item (see Section 4.4).

11.2. Threat: On-Path Modification

The SAI values are necessary to correlate EDHOC messages and to decrypt SAFE PDUs, so must be in plaintext and any on-path attacker modification of these values will cause either the EDHOC negotiation to fail or the SAFE decryption to fail.

The IV values are already part of what would be sent with the associated ciphertext, and any modification will cause the SAFE decryption to fail.

Both of these conditions are detectable by the receiver being attacked so there is no possibility of the attacker causing undetectable changes.

11.3. Threat: Denial of Service

The behaviors described in this section all amount to a potential denial-of-service to a participating node. The denial-of-service could be limited to an individual node, or could affect all entities on a host or network segment.

// TBD

12. IANA Considerations

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

12.1. Well-Known IPN Service

Within the registry group of [IANA-URI], the registry titled "'ipn' Scheme URI Well-known Service Numbers for BPv7" has been updated to include the following entry.

Value	Description	Reference
TBA3	BPv7 SAFE Messaging	[This specification]

Table 16: 'ipn' Scheme URI Well-known Service Numbers for BPv7

12.2. EDHOC Registries

Within the registry group of [IANA-EDHOC], the registry titled "EDHOC Exporter Labels" has been updated to include the following entry.

Label	Description	Reference
TBA4	Derived BPv7 SAFE Secret	[This specification]

Table 17: EDHOC Exporter Labels

Within the registry group of [IANA-EDHOC], the registry titled "EDHOC External Authorization Data" has been updated to include the following entry.

Name	Label	Description	Reference
BPv7 SAFE	TBA5	Set of Create messages for BPv7 SAFE	[This specification]

Table 18: EDHOC External Authorization Data

12.3. BP SAFE Registries

A new registry group "Bundle Protocol (BP) Security Associations with Few Exchanges (SAFE)" has been created at [IANA-SAFE].

Within the registry group of [IANA-SAFE], the registry titled "SAFE Security Modes" has been created with registration procedures from Table 19 and initial contents of Table 20.

Range	Registration Procedures
-32768 to -32641	Experimental use
-32640 to -1	Private use
0 to 255	RFC Required
256 to 32767	Specification Required

Table 19: SAFE Security Modes Registration

Value	Name	Reference
-32768 to -32641	Reserved for experimental use	[This specification]
-32640 to -1	Reserved for private use	[This specification]
0	_reserved_	[This specification]
1	end-to-end	[This specification]
2	one-hop	[This specification]
3 to 32767	_Unassigned_	

Table 20: SAFE Security Modes

Within the registry group of [IANA-SAFE], the registry titled "SAFE Activity Types" has been created with registration procedures from Table 21 and initial contents of Table 22.

Range	Registration Procedures
-32768 to -32641	Experimental use
-32640 to -1	Private use
0 to 255	RFC Required
256 to 32767	Specification Required

Table 21: SAFE Activity Types Registration

Value	Description	Notation	Reference
-32768 to -32641	Reserved for experimental use		[This specification]
-32640 to -1	Reserved for private use		[This specification]
0	Reserved for Initial Authentication	IA	Section 5.1 of [This specification]
1	Capability Indication	CI	Section 5.2 of [This specification]
2	SA Creation	SC	Section 5.3 of [This specification]
3	SA Rekey	SR	Section 5.4 of [This specification]
4	SA Teardown	ST	Section 5.5 of [This specification]
5	Event Notification	EN	Section 5.6 of [This specification]
6 to 32767	_Unassigned_		

Table 22: SAFE Activity Types

Within the registry group of [IANA-SAFE], the registry titled "SAFE Message Data Items" has been created with the following initial contents. Each entry in the table indicates which messages are valid for containing the associated data item and the map label by which the item is identified. The registration procedure for this registry is identical to the corresponding "SAFE Activity Types" entry from the first column.

This registry includes only well-known data item types; private use labels can be used to include any other data items in a message.

Activity Type Notation	Label	Description	Notation	Reference
all types	-32768 to -1	_Reserved for private use_		[This specification]
all types	0	Error Type Enumeration	ETE	Section 6.1 of [This specification]
IA		_this activity does not use data item labels_		
CI	1	Concurrent Activity Support	CAS	Section 6.2 of [This specification]
CI	2	EID Scheme Support	ESS	Section 6.3 of [This specification]
CI	3	BPSec Context Support	BCS	Section 6.4 of [This specification]
SC	1	Security Association Identifier	SAI	Section 6.5 of [This specification]
SC	2	Additional Key Exchange	AKE	Section 6.6 of [This specification]
SC	3	Additional Random Nonce	ARN	Section 6.7 of [This specification]
SC	9	Security Mode Selector	SMS	Section 6.8 of [This specification]

SC	6	Endpoint Selector at Initiator	ESI	Section 6.10 of [This specification]
SC	7	Endpoint Selector at Responder	ESR	Section 6.10 of [This specification]
SC	8	Validity Node Time Interval	NTI	Section 6.9 of [This specification]
SC	4	Security Operation Selector	SOS	Section 6.11 of [This specification]
SC	5	Key Use Selector	KUS	Section 6.12 of [This specification]

Table 23: SAFE Message Data Items

Within the registry group of [IANA-SAFE], the registry titled "SAFE Error Type Enumerations" has been created with the following initial contents. These entries are distinct and separate from the "EDHOC Error Codes" registry of [IANA-EDHOC].

Activity Type Notation	Value	Description	Reference
all types	-32768 to -32641	Reserved for experimental use	[This specification]
all types	-32640 to -1	Reserved for private use	[This specification]
all types	1	Invalid data item label	[This specification]
all types	2	Unknown data item label	[This specification]
all types	3	Invalid data item value	[This specification]
SR,ST	256	Unknown security association	[This specification]

Table 24: SAFE Error Type Enumerations

13. References

13.1. 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-EDHOC]
IANA, "Ephemeral Diffie-Hellman Over COSE (EDHOC)",
<https://www.iana.org/assignments/edhoc/>.

[IANA-SAFE]
IANA, "Bundle Protocol (BP) Security Associations with Few Exchanges (SAFE)",
<https://www.iana.org/assignments/bp-safe/>.

[IANA-URI] IANA, "Uniform Resource Identifier (URI) Schemes",
<https://www.iana.org/assignments/uri-schemes/>.

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC9528] Selander, G., Preu Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9528, DOI 10.17487/RFC9528, March 2024, <https://www.rfc-editor.org/info/rfc9528>.

[I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., Hglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-13, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-13>.

[I-D.ietf-dtn-bibect]
Burleigh, S., Montilla, A., Deaton, J., and C. Caini, "Bundle-in-Bundle Encapsulation", Work in Progress, Internet-Draft, draft-ietf-dtn-bibect-05, 15 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-dtn-bibect-05>.

[I-D.ietf-dtn-bpsec-cose]
Sipos, B., "Bundle Protocol Security (BPSec) COSE Context", Work in Progress, Internet-Draft, draft-ietf-dtn-bpsec-cose-08, 3 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-dtn-bpsec-cose-08>.

[I-D.ietf-dtn-eid-pattern]
Sipos, B., "Bundle Protocol Endpoint ID Patterns", Work in Progress, Internet-Draft, draft-ietf-dtn-eid-pattern-02, 13 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-dtn-eid-pattern-02>.

13.2. Informative References

[802.1X] IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Port-Based Network Access Control", IEEE 802-1x-2020, DOI 10.1109/IEEESTD.2020.9018454, 28 February 2020, <https://ieeexplore.ieee.org/document/9018454>.

[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>.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <https://www.rfc-editor.org/info/rfc4838>.

[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>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[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>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[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>.

[RFC9529] Selander, G., Preu Mattsson, J., Serafin, M., Tiloca, M., and M. Vuini, "Traces of Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9529, DOI 10.17487/RFC9529, March 2024, <https://www.rfc-editor.org/info/rfc9529>.

Appendix A. Example Sequencing

This section contains a minimal example of two nodes initializing a primary SA and two secondary SAs with overlapping activities using pipelined messaging. The EDHOC messages in the IA activity are taken from the example in Section 2 of [RFC9529] to avoid duplicating EDHOC example logic here. This example does not include any PDU losses so no retransmission is needed.

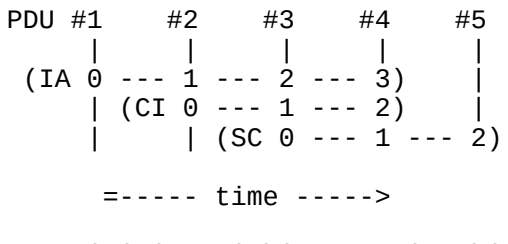


Figure 19: Activities within PDUs in this example

A.1. SAFE PDU #1

The encoded PDU #1 (initiator-to-responder) is the following byte string:

```
01f6f50006582031f82c7b5b9cbbf0f194d913cc12ef1532d328ef32632a4881a1c0
701e237f042d
```

The contents of that PDU are following CBOR sequence:

```
1, / version /
null, / partial-iv /
true, / rx-sai /
/ message_1: /
0, / method /
6, / suites /
h'31f82c7b5b9cbbf0f194d913cc12ef1532d328ef32632a48
  81a1c0701e237f04', / G_X /
-14 / C_I /
```

A.2. SAFE PDU #2

The encoded PDU #2 (responder-to-initiator) is following byte string:

```
01f62d5880dc88d2d51da5ed67fc4616356bc8ca74ef9ebe8b387e623a360ba480b9
b29d1c9e2115f057ff0687a41a21a1d7fc4553061357fe93dbef391c83e18a27b29c
e24e44a45916fdebe9c5a505abdec132750551154966a67eed27236c093e2774814f
11c5affadf8c0cd96e31fef7deda3398136d3e79419efd86efe646e1606b77
```

That PDU has the following decoded header, eliding the ciphertext message.

```
1, / version /
null, / partial-iv /
-14 / rx-sai from C_I /
/ message_2: h'dc88...6b77' /
```

Derived from the shared secrets, the PLAINTEXT_2 contents are the following:

```
4118A11822822E4879F2A41B510C1F9B58407934E54041AE5ACB7C68DE7ED477D947
E4214E84659D99FCD49C4B25033DD92AF50DE0570BC27B72039CEB2F4A4F8F05082F
F9739A823C25634FE42BB306F296364F010001A3011904000282010203810 3
```

That plaintext has the following decoded items.

```
h'18', / C_R /
{34: [-15, h'79F2A41B510C1F9B']}, / ID_CRED_R with following sig. /
h'A9CAAE6BB615B3C50D774AFA0B6297AA3CCDBED1A03A34614DEE8A62216F685DC9
  ED1C8DE7291E2C571E1E9DAD39B350869CC7B2658514EB7881E8B03D5C9D89',
-23, / EAD label with following value /
h'010001A3011904000282010203810 3'
```

The embedded EAD_2 contains the following SAFE message:

```
1, / act. index /
0, / act. step /
1, / act. type: CI /
{
    1: 1024, / CAS: max 1024 /
    2: [1, 2], / ESS: dtn and ipn schemes /
    3: [3] / BCS: COSE Context /
}
```

A.3. SAFE PDU #3

The encoded PDU #3 is following byte string:

```
01f6411858c1e418256972ed10356c2cadc3e72dd86c16e830590972a7fa238c92b3
0c7cb6a030397061de4327b02b5aae7c0e1c6de2d67bd7627b33b8cb8e9bb8303aca
53143aacd8713413becc1a3ad2ad7ac3bee1b56cb576637738787106a415c41ab3c4
b70fe87782994bb93d25c2a7565255948267637511364b2236ae8ddbbfb4321246fc
38ba317216893f9afe59b85a4301e871944d8e3afb7de0278bf2cf7c161ce6fe2ff6
b2e6d8f5666d5ae4c555f07ee534293f4dc85d282bdd35e2676b126eee
```

That PDU has the following decoded header, eliding the ciphertext message.

```
1, / version /
null, / partial-iv /
h'18' / rx-sai from C_I /
/ message_3: h'e418...6eee' /
```

Derived from the shared secrets, the PLAINTEXT_3 contents are the following:

```
A11822822E48C24AB2FD7643C79F5840A6F33C8F1FED68F30F83261652D193DC1B29
A5B40A53AA6D695466F8E473B74548914C293F8EB127C71813993338E6573F90A60A
BB6F3484FE4172056B645A0A36584D010002A80146235A91D189EA035056B44D9A0F
87538A24CA8EBE49D4FD51040305A10101090106F607F602582031F82C7B5B9CBBF0
F194D913CC12EF1532D328EF32632A4881A1C0701E237F04364F010101A301190400
02820102038103
```

That plaintext has the following decoded items.

```
{34: [-15, h'C24AB2FD7643C79F']}, / ID_CRED_I with following sig. /
h'A6F33C8F1FED68F30F83261652D193DC1B29A5B40A53AA6D695466F8E473B74548
  914C293F8EB127C71813993338E6573F90A60ABB6F3484FE4172056B645A0A',
-23, / EAD label with following value /
h'010002A80146235A91D189EA035056B44D9A0F87538A24CA8EBE49D4FD51040305
  A10101090106F607F602582031F82C7B5B9CBBF0F194D913CC12EF1532D328EF32
  632A4881A1C0701E237F04',
-23, / EAD label with following value /
h'010101A30119040002820102038103'
```

The embedded EAD_3 contains the following SAFE messages:

```
1, / act. index /
0, / act. step /
2, / act. type: SC /
{
    1: h'235A91D189EA', / local SAI bytes /
    3: h'56B44D9A0F87538A24CA8EBE49D4FD51', / ARN bytes /
    4: 3, / CTX: COSE (3) /
    5: { / KUS: /
      1: 1 / alg: A128GCM (1) /
    },
    9: 1, / SMS: end-to-end (1) /
    6: null, / TSI: still TBD /
    7: null, / TSR: still TBD /
    2: h'31F82C7B5B9CBBF0F194D913CC12EF1532D328EF32632A4881A1C0701E2
         37F04' / AKE bytes /
}
```

```
1, / act. index /
1, / act. step /
1, / act. type: CI /
{
    1: 1024, / CAS: max 1024 /
    2: [1, 2], / ESS: dtn and ipn schemes /
    3: [3] / BCS: COSE Context /
}
```

A.4. SAFE PDU #4

The encoded PDU #4 is following byte string:

```
01F62D586492459A97A617FE0176E1040FB1E98A640FBDF104E4839EB576F32EF07C
F283B9C8FF4AC2B5B3DC72F15A5CAEC383A366A8310D97E369E1696D4D43BC9BAA99
E63E750D29344A95D9C703B21B7AE796F36F82FDF8798F5428AD8057C2D197B7F6DE
7ADEB6
```

That PDU has the following decoded header, eliding the ciphertext message.

```
1, / version /
null, / partial-iv /
-14 / rx-sai from C_I /
/ message_3: h'9245...DEB6' /
```

Derived from the shared secrets, the PLAINTEXT_4 contents are the following:

```
3642010236584d010102a80146a8c046494ebd035043184c4d9f379d2eb35fd2f2f1
1ae27b040305a10101090106f607f6025820dc88d2d51da5ed67fc4616356bc8ca74
ef9ebe8b387e623a360ba480b9b29d1c
```

That plaintext has the following decoded items.

```
-23, / EAD label with following value /
h'0102',
-23, / EAD label with following value /
h'010102A80146A8C046494EBD035043184C4D9F379D2EB35FD2F2F11AE27B040305
  A10101090106F607F6025820DC88D2D51DA5ED67FC4616356BC8CA74EF9EBE8B38
  7E623A360BA480B9B29D1C'
```

The embedded EAD_4 contains the following SAFE messages:

```
1, / act. index /
2 / act. step /
```

```
   1, / act. index /
   1, / act. step /
   2, / act. type: SC /
   {
       1: h'A8C046494EBD', / local SAI bytes /
       3: h'43184C4D9F379D2EB35FD2F2F11AE27B', / ARN bytes /
       4: 3, / CTX: COSE (3) /
       5: { / KUS: /
           1: 1 / alg: A128GCM (1) /
       },
       9: 1, / SMS: end-to-end (1) /
       6: null, / TSI: still TBD /
       7: null, / TSR: still TBD /
       2: h'DC88D2D51DA5ED67FC4616356BC8CA74EF9EBE8B387E623A360BA480B9B
            29D1C' / AKE bytes /
   }
```

A.5. SAFE PDU #5

The encoded PDU #5 is following byte string:

```
01410141185306684BF319BBE5B2237EF71606DB2714E0F992
```

That PDU has the following decoded header, eliding the ciphertext message.

```
1, / version /
h'01', / partial-iv /
h'18' / rx-sai from C_R /
/ message_3: h'0668...F992' /
```

Derived from the shared secrets, the PLAINTEXT contents are the following:

```
420102
```

That plaintext has the following decoded items.

```
h'0102'
```

The embedded byte strings contain the following SAFE messages:

```
1, / act. index /
2 / act. step /
```

Acknowledgments

Thanks go to Leonardo Babun and Cherita Corbett of JHU/APL for pre-draft review and editing of this document.

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