Delay-Tolerant Networking
Internet-Draft
Intended status: Standards Track
Expires: 12 October 2026

B. Sipos
JHU/APL

10 April 2026

Bundle Protocol (BP) Security Associations with Few Exchanges (SAFE)
draft-sipos-dtn-bp-safe-01

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. This protocol also provides for unilateral re-keying of established security associations in a delay-tolerant manner.

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 12 October 2026.

Copyright Notice

Table of Contents

1. Introduction 4
1.1. Scope 6
1.2. Use Cases Considered 7
1.2.1. End-to-End Security 8
1.2.2. One-Hop Transport Security 9
1.2.3. Multi-Hop Transport Security 10
1.2.4. Tunnel Security 11
1.3. Use of CDDL 11
1.4. Terminology 11
2. General Protocol Description 13
2.1. Content Key Lifecycle Management 15
2.2. Relationship to EDHOC 17
2.3. Relationship to BPSec 17
2.4. Credential Sources 18
2.5. Extensibility 18
2.5.1. EDHOC Cipher Suites 19
2.5.2. EDHOC Methods 19
2.5.3. EDHOC Credential Types and Identifiers 19
2.5.4. SAFE Activity Types and Data Items 20
2.5.5. BPSec Contexts and Options 20
3. Information Bases 20
3.1. Participating Peers 21
3.2. Activity States 22
3.3. Primary Security Associations 24
3.4. Secondary Security Associations 29
4. Protocol Sub-Layers and Binding 31
4.1. Activity Sequencing 33
4.2. Message Structure 34
4.2.1. Payload Data Map 36
4.2.2. Error Indication 37
4.3. Activity States, Deduplication, and Retransmission . . . 37
4.4. Message Aggregation 40
4.5. Packetization 41
4.6. PDU Transport 42
5. Activity Types 43
5.1. Initial Authentication (IA) 43
5.1.1. EDHOC Requirements 45

5.2. Capability Indication (CI) 45
5.3. Event Notification (EN) 47
5.4. SA Creation (SC) 47
5.5. SA Teardown (ST) 50
5.6. Key Creation (KC) 51
5.7. Key Discard (KD) 53
5.8. Key Reject (KR) 54
5.9. Prekey Creation (PC) 55
5.10. Prekey Discard (PD) 56
5.11. Prekey Reject (PR) 57
6. Data Item Types 58
6.1. Concurrent Activity Support (CAS) 59
6.2. EID Scheme Support (ESS) 59
6.3. BPSec Context Support (BCS) 60
6.4. Key Depth Range (KDR) 60
6.5. Security Association Identifier (SAI) 60
6.6. Key Identifier (KID) 61
6.7. Initial Content Key (ICK) 61
6.8. Public Key Structure (PKS) 61
6.9. Additional Random Nonce (ARN) 62
6.10. Security Mode Selector (SMS) 62
6.11. Node Time Interval (NTI) 62
6.12. Endpoint Selectors (ESx) 63
6.13. Security Operation Selectors (SOS) 63
6.14. Key Use Selectors (KUS) 64
7. Activity Patterns 64
7.1. Concurrent Activities 65
7.2. Establishment Ordering 67
7.3. Content Key Upkeep 69
7.4. Prekey Upkeep 69
8. Shared Secret Derivations 69
8.1. Security Association PRK 71
8.2. Prekey Shared Secret 72
8.3. Content Key PRK 72
8.4. Primary SA Content Keys 73
8.5. Secondary SA Content Keys 74
9. Security Association Uses 74
9.1. SAFE Confidentiality 75
9.1.1. Without a Primary SA 75
9.1.2. With a Primary SA 76
9.2. Common BPSec Operation 78
9.3. Binding for BIB-HMAC-SHA2 Context 79
9.3.1. Secondary SA Information 80
9.3.2. KUS Options 80
9.3.3. Key Derivation 81
9.3.4. BPSec Operation 81
9.4. Binding for BCB-AES-GCM Context 82
9.4.1. Secondary SA Information 82

9.4.2. KUS Options 82
9.4.3. Key Derivation 83
9.4.4. BPSec Operation 83
9.5. Binding for COSE Context 84
9.5.1. Secondary SA Information 84
9.5.2. KUS Options 85
9.5.3. Key Derivation 86
9.5.4. BPSec Operation 87
10. PKIX Certificate Profile 91
11. Security Considerations 91
11.1. Threat: Passive Leak of Data 91
11.2. Threat: On-Path Modification 91
11.3. Threat: Denial of Service 92
12. IANA Considerations 92
12.1. Well-Known IPN Service 92
12.2. EDHOC Registries 92
12.3. BP SAFE Registries 93
13. References 100
13.1. Normative References 100
13.2. Informative References 103
Appendix A. Example Sequencing 105
A.1. SAFE PDU #1 105
A.2. SAFE PDU #2 105
A.3. SAFE PDU #3 106
A.4. SAFE PDU #4 108
A.5. SAFE PDU #5 110
Acknowledgments 111
Implementation Status 111
Author’s Address 111

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 target 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 described below.

Independent Asymmetric Operations: The security source can use a static 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.

Unmanaged Derived Keys: For key types which support key exchange, such as elliptic curve (EC) keys, the security source can use a static 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 cipher suites 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 provide any authentication of the peer entity (via some identity binding to the associated private key). It also does not allow for forward secrecy (FS) or post-compromise security (PCS) because the original asymmetric keys need to be long-lived enough to handle all communications between the entities (_i.e._, considered "static" keys).

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" control plane 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.

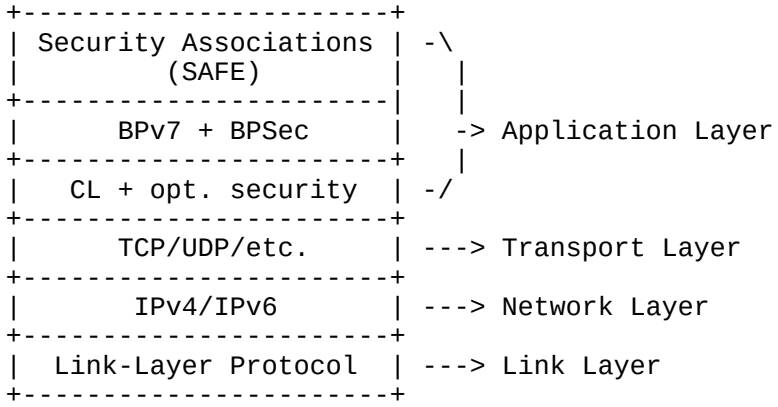


Figure 1: The Locations of SAFE and BP above the Internet Protocol Stack

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 adapter(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.

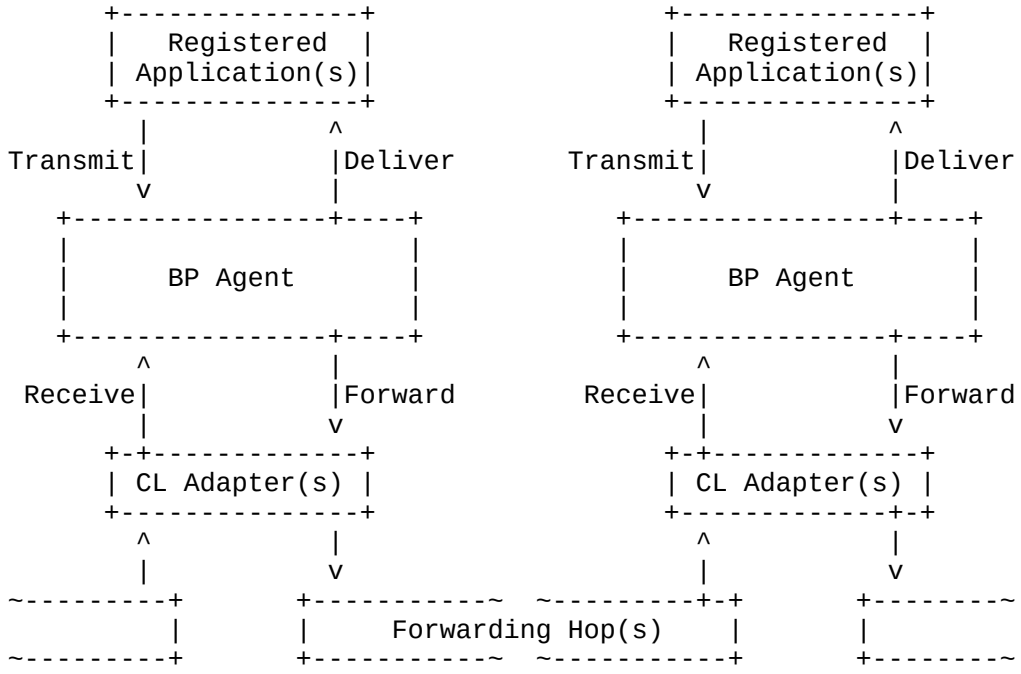


Figure 2: BP Agent data plane interaction points (showing two nodes)

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 Section 6.10.

For this case 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).

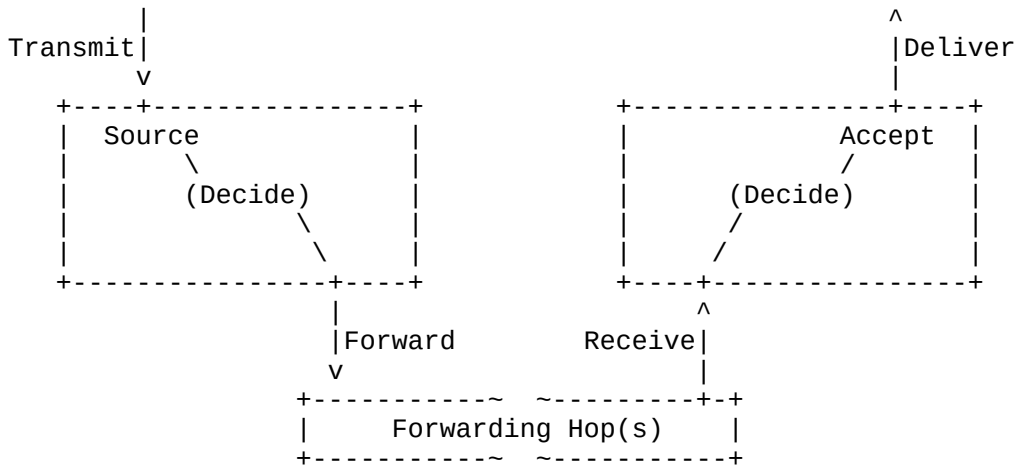


Figure 3: End-to-End security flows

The purpose of the mode in this use case is to ensure that selected 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. For that reason, the selected traffic SHOULD include only bundles with source and destination endpoints on those nodes.

Because the security operation here has the maximal lifetime, its target needs to be another block that is guaranteed to exist from bundle creation up to the point of delivery. Because of that, the target block (and any other covered blocks) for the end-to-end case SHALL to be one that also exists for the lifetime of the bundle.

1.2.2. One-Hop Transport 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 transport mode of Section 6.10.

For this case 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.

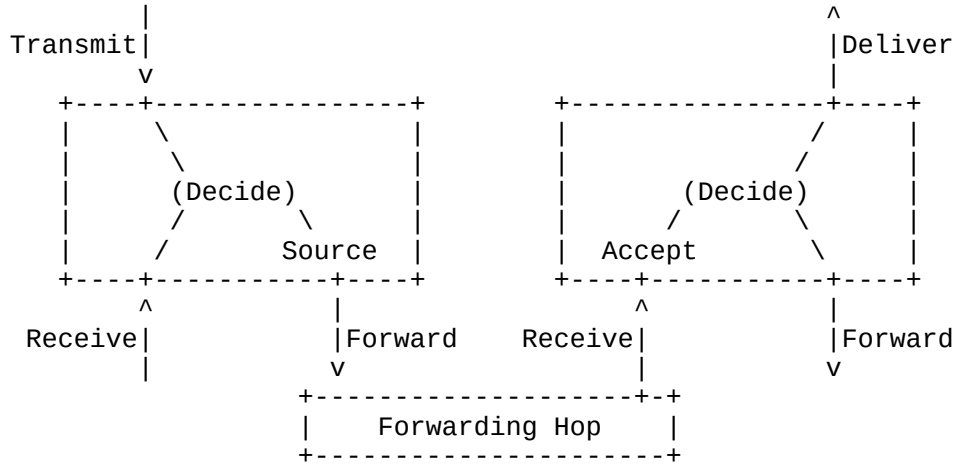


Figure 4: One-Hop security flows

The purpose of the mode in this use case is to supplement security mechanisms (if any) provided by the forwarding convergence layer (CL) protocol stack. 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. For that reason, the selected traffic SHOULD include all bundles between the participating nodes. A side effect of using SAFE in this mode, as a supplement to CL security, is that it can normalize the security capability between the two nodes because it operates above and independently from whichever CL types are used between them (even if that CL type or its properties change over time).

Because the security operation here has a limited lifetime, its target could be any block in the bundle. However, the constraint of Section 3.2 of [RFC9172] requires that the combination of security service and target block be unique within the bundle. Because of that, the target block for the one-hop case SHOULD to be one that also has a one-hop lifetime such as a Previous Node block.

1.2.3. Multi-Hop Transport Security

The multi-hop use case is where the SAFE entities negotiate secondary SAs which match arbitrary endpoints but apply to bundles traversing a several hops between the participating nodes. This use case corresponds to the transport mode of Section 6.10, just as the one-hop use case of Section 1.2.2.

For this case the bundle flows and security processing will look similar to that of Figure 4 except that there are more hops between the participating nodes. The negotiated security service is sourced immediately before forwarding (specifically to the peer node of the SA), has a lifetime of several bundle hops, and is accepted immediately upon reception at that peer node.

Also similar to the one-hop case, there are more constraints and considerations for the target of the security operation. One consideration is the fact that BP topology or routing, outside the scope of this protocol, SHALL be configured to ensure that the selected traffic actually gets forwarded between the participating nodes. If the actual forwarding path passes through one of the nodes but not the other, the security operation will be sourced properly but never accepted and will serve no purpose.

Constraints on the target for this use case are more complex than for the one-hop case because the security operation needs to ensure uniqueness for its lifetime while also not targeting blocks with one-hop lifetime (or blocks which are expected to be modified at each

hop). This use case could include integrity operations targeting an extension block specific to a transit network, where the extension (and its security) is added at ingress to the transit network and is removed at egress from the transit network.

1.2.4. Tunnel Security

// TBD a use case that configures BPSec as well as BIBE to require
// encapsulation of selected traffic between the participating nodes.

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). These bundles are communicated between SAFE entities on participating nodes.

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

SAFE entity: An implementation artifact which manages the SAFE states defined in this specification within each participating node.

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 content keys.

Content Key (CK): An individual symmetric key material along with metadata controlling and authorizing its use within a specific context. One context for CK use is for confidentiality of SAFE messages between SAFE entities. Another context for CK use is within the data plane of participating nodes outside of the SAFE entities.

Activity: Each SAFE activity consists of a sequence of messages being exchanged, via protocol messaging, between two participating entities to achieve a specific narrow goal. Each activity instance a specific type and one SAFE entity as its initiator. Each type of activity consists of a well-defined number of steps of messaging between its entities.

Message: Each step of an activity is realized by a message identifying its activity index and step number. Each data-bearing message also identifies its activity type and contains a set of type-specific 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.

Unilateral activity: An activity type which consists of a single data-bearing message from the initiator followed by a single acknowledgement message from the responder. There is no negotiated state for these activity types, and the acknowledgement is purely for retransmission sub-layer needs.

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 non-SAFE data plane traffic as described in Section 2.3. To avoid coupling each SAFE entity with any BPSec entity on the same node, SAFE PDUs use application-provided security instead of relying on BPSec.

The rough order of protocol operation is as follows:

1. Establish a confidential channel between a pair of SAFE entities using ephemeral keys.

2. Perform mutual authentication based on a trusted credential for each entity.

3. Exchange the capabilities and constraints from each entity.

4. Negotiate a primary SA for SAFE PDUs, with initial pair of content keys for SAFE PDUs.

5. Negotiate some number of secondary SAs for specific BPSec operations on specific traffic.

6. Unilaterally upkeep content keys for primary and secondary SAs over time, with optional FS/PCS for each new key.

7. Create or teardown secondary SAs as BPSec needs change over time.

8. Re-authenticate or teardown the primary SA as determined by policy.

The initial confidential channel between SAFE entities is provided by an Ephemeral Diffie-Hellman Over COSE (EDHOC) session [RFC9528] and Section 2.2, which includes forward secrecy, mutual authentication, and proof-of-possession. EDHOC also allows using external authorization data (EAD) items to confidentially transport SAFE messages during session establishment (see Section 9.1.1), which avoids excess delay before negotiating secondary SAs.

A side effect of the EDHOC session is the establishment of a primary SA between the two SAFE peers and an initial pair of content keys for that primary SA. The primary SA keys provide SAFE message confidentiality after the initial authentication EDHOC session (see Section 9.1.2). 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, both primary and secondary, including life-cycle management of subordinate symmetric keys (see Section 2.1), in order to bound key use over time and over volume of processed data.

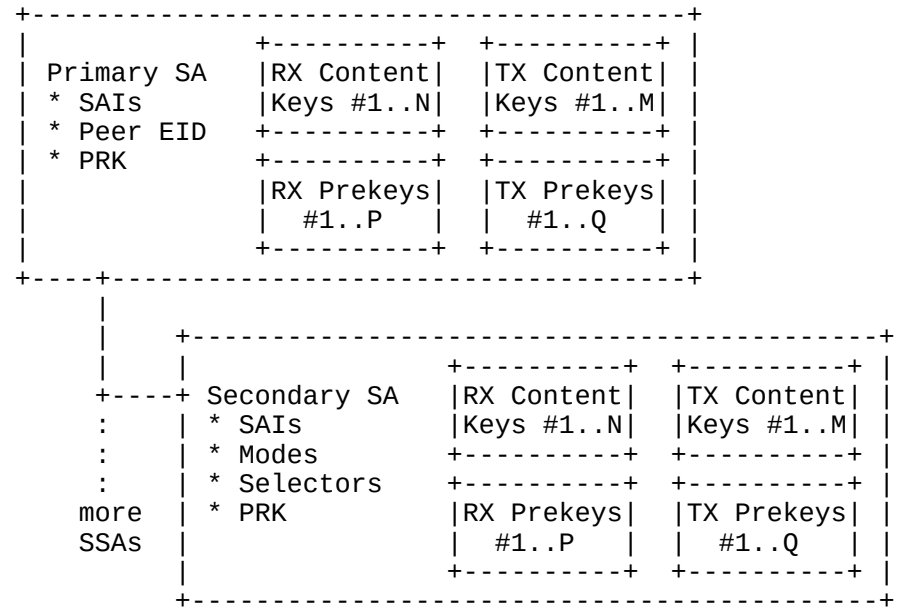


Figure 5: Relationships between SAs and their key material

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 (PBNAC) 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 (Section 4) is organized to avoid a sequence-of-exchanges pattern and its activities (Section 5) are organized to minimize the number of steps (_i.e._, round trips) in each.

Instead of a two-way request--response pattern where all retransmission occurs from the requesting entity, the SAFE pattern is an activity sequence where each step acknowledges the receipt of the previous step and 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 results in more total messages than two-way exchanges, but it enables pipelining of messages and selective retransmission (ARQ) of individual encoded messages from either side of an activity conversation.

Some SAFE activities involve negotiating of a shared state between an initiator and a responder using two data-bearing steps and a final acknowledgement. Many activities are unilateral, meaning they consist of a single data-bearing step from the initiator followed by a single responder acknowledgement. Having activities be unilateral makes the overall protocol more delay-tolerant.

2.1. Content Key Lifecycle Management

Within each SA, both primary and secondary, there are some number of content keys available for use in the data plane. The primary SA content keys are used for AEAD of SAFE PDUs themselves, while secondary SA content keys are for BPSec operations on other traffic.

The primary SA has an implicit pair of content keys created as part of the Initial Authentication (IA) activity, each with a key identifier (KID) derived from and the same length as the primary security association identifier (SAI). All other KIDs of primary and secondary SAs are explicitly chosen by the entity which would use that key as the security source (_i.e._, the TX side of the data plane).

Within a single SA, the lifecycle of a single content key is depicted in Figure 6 where "Entity A" is the TX end of that key and "Entity B" is the RX end. The optional Prekey Creation activity and its public key data is used to ensure forward secrecy of the content key when desired by the TX end. The Key Discard activity is used to inform the RX end when the content key will no longer be used for security operations and it is safe to be discarded.

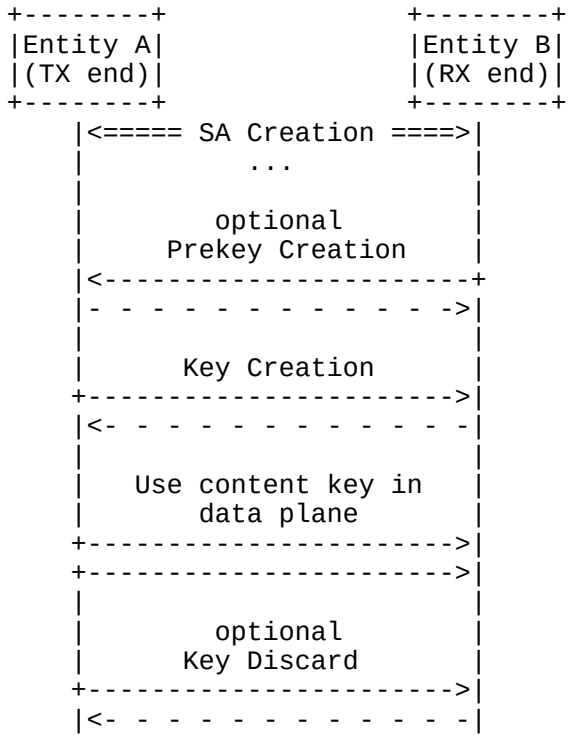


Figure 6: Activity sequence for one SA content key

An important aspect that distinguishes security associations from their constituent content keys is that the SA is parameterized to determine which traffic (either SAFE PDUs or data plane traffic) is covered by security and how that security is applied but does not contain any intrinsic time-variance, while each CK is parameterized to have a single validity time interval bounding its use over wall-clock time. This means that the SA lifecycle is managed in wall-clock time by activities between SAFE entities, specifically SA Creation (SC) and SA Teardown (ST). Within each SA, the CKs intrinsic time-variance allows pre-planning a sequence of keys with future validity to enable continuous operation by rolling-over content keys as they expire in wall-clock time as illustrated in Figure 7 for a single data direction. All of those CKs can be created immediately after SC creation with future validity times.

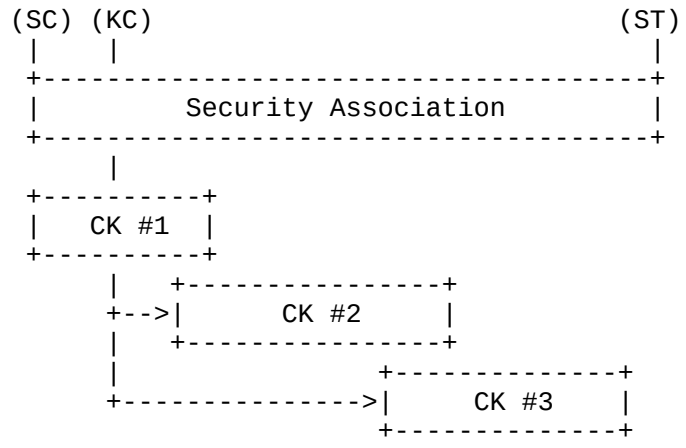


Figure 7: Timeline of SA content keys

2.2. 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 as critical EAD items. 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) using EDHOC session parameters.

2.3. Relationship to BPSec

This SAFE protocol functions as a control plane for establishing secondary SAs, which are then used to provide symmetric content keys to be used by BPSec for data plane security operations on individual target blocks within specific bundles as described in Section 1.2. The scoping of each secondary SA includes a BPSec security context identifier and its options for which each derived content key is authorized to be used. It is up to BPSec policy on each of the SAFE peer nodes to enforce those authorized uses as the mandatory security source and mandatory security acceptor for matching flows in each direction.

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 each node.

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 with the chosen security service.

2.4. 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.

2.5. 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.

2.5.1. EDHOC Cipher Suites

The EDHOC cipher suite, chosen by the initiator and agreed with the responder, controls the cryptographic algorithms used to secure the EDHOC session, derive internal key material, authenticate each entity to its peer, and to secure SAFE messages after the EDHOC session has completed. The base specification in Section 3.6 of [RFC9528] defines 9 cipher suites which cover capabilities all the way from lightweight algorithms targeting constrained processors to one which conforms to the US CNSS Commercial National Security Algorithm Suite (CNSA) 1.0 [CNSA1].

Additional cipher suites [I-D.spm-lake-pqsuites] enable the use of ML-KEM for EDHOC session confidentiality and ML-DSA for signature-based entity authentication which
// do not yet conform to CNSA 2.0 [CNSA2]. These cipher suites update the semantics of some fields in the EDHOC messages but not their structure or encoding.

2.5.2. EDHOC Methods

The EDHOC method controls how each entity authenticates to its peer within an EDHOC session. The base specification in Section 3.2 of [RFC9528] defines four methods to cover authenticating each side with either an asymmetric-key signature or an elliptic-curve Diffie-Hellman (ECDH) derived-key MAC.

Additional methods [I-D.pocero-authkem-edhoc] enable the use of ML-KEM for efficient MAC-based authentication in a way which
// do not yet conforms to CNSA 2.0. These methods alter some behaviors related to the internal key schedule without affecting the names or uses of derived PRK values.

2.5.3. EDHOC Credential Types and Identifiers

The credential type used to authenticate each entity and how it is identified to its peer are both chosen by that entity being authenticated, based on a profile exposed outside of this protocol. The credential identifiers are derived from COSE header parameters from [IANA-COSE] and currently include a X.509 certificate thumbnail, URL, or certificate chain data, or a C509 certificate thumbnail, URL, or certificate chain data. Future credential types are expected to include pre-shared key material.

2.5.4. SAFE Activity Types and Data Items

The activity types defined in this document are expected to be sufficient for SA creation and management, but there is a registry of available types defined in Section 12.3 for future expansion as necessary including reserved blocks for private and experimental uses.

Within each activity type, specifically the SA Creation (SC) type, there is a block of data item code points reserved for private use and a block left unassigned for future specifications. These are expected to include additional selectors for traffic flows beyond just source and destination EID from Section 6.12.

2.5.5. BPSec Contexts and Options

There are currently only two well-known BPSec contexts defined in [RFC9173] and one drafted in [I-D.ietf-dtn-bpsec-cose]. The first pair from [RFC9173] are intended to be "... for testing the interoperability ... and for providing basic security operations when no other security contexts are defined or otherwise required for a network" and these have few options needed to operate them as described in Section 9.3 and Section 9.4.

The COSE context from [I-D.ietf-dtn-bpsec-cose] is more full-featured, but that also means it has more options and more complex configurations of those options available as described in Section 9.5; COSE itself is also expected to be expanded in the near- and long-term to include PQC-type algorithms.

Future contexts are expected to be specialized to specific missions and networks. Well-known contexts will be registered with IANA under [IANA-BUNDLE] and need to define a SAFE binding if they are to be used with negotiated secondary SAs.

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 entities.

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.
Properties below are based on the above key column.	
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 end-entity certificate
* x5t (34), x5u (35), x5chain (33) to identify an X509 end-entity certificate
* 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 10 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 for the activity, which is unique to and defined by the initiator.
Properties below are based on the above key columns.	
Last Step Transmitted (LTX)	The step of the activity which is associated with the last message sent to the peer.
Last Message Transmitted	The logic of activity step ARQ requires messages to be re-transmitted with the same content as the original. One way this can be controlled by each entity is to simply retain an encoded form of each TX message until it is acknowledged by the subsequent activity step. The actual mechanism of retaining and re-transmitting each message is implementation defined.
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, as depicted in Figure 5. The state of each primary SA known to the local entity has logical columns as indicated in Table 3.

A primary SA covers traffic in both directions between the two peers participating in the SA. The primary SA has no specific endpoint selectors because each is used for security between the SAFE entities themselves rather than any other, data plane traffic.

Name	Description
Local SAI	The locally-generated SA identifier for this entry. This Local SAI alone (within the local SAFE entity) is a unique identifier for the primary SA.
Peer EID	The transport EID for the SAFE entity on the peer node.
Peer SAI	The peer-generated SA identifier for this entry. The combination of Peer EID and Peer SAI together form a unique identifier for the primary SA.
Properties below are based on the above key columns.	
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_EXT and SAFE_EXP functions defined in Section 8.
Primary pseudo-random key (PRK):	Internal key material derived from the EDHOC session for this SA, as the byte string PRK_SA defined in Section 8.1. This is used to derive further values for content key material as defined in Section 8.4.

TX Content Key Depth Range:	The agreed range of valid size of the TX Content Key store for this SA, which is the intersection of the local and received range. New content keys will be generated as needed to stay within this range.
TX Content Keys	The primary SA has one or more content keys for outgoing PDUs to the Peer EID. Each content key has the information defined in Table 4 and is scoped within a single primary SA. The associated key material is derived from the shared secret created during Initial Authentication (IA) and Key Creation (KC) activities.
RX Content Key Depth Range:	The agreed range of valid size of the RX Content Key store for this SA, which is the intersection of the local and received range. New content keys will be accepted only if the size would stay within this range.
RX Content Keys	The primary SA has one or more content keys for incoming PDUs from the Peer EID. Each content key has the information defined in Table 5 and is scoped within a single primary SA. The associated key material is derived from the shared secret created during an Initial Authentication (IA) activity.
RX Prekey Depth Range:	The agreed range of valid size of the RX Prekey store for this SA, which is the intersection of the local and received range. New prekeys will be generated as needed to stay within this range.
RX Prekeys	A set of one-time-use private asymmetric key material used to process received CK Creation activities. These private keys are generated locally and their associated public prekeys are communicated to the peer entity via CK Prekey activities. Each private prekey has the information defined in Table 6 and is scoped within a single primary SA.
TX Prekey Depth Range:	The agreed range of valid size of the TX Prekey store for this SA, which is the intersection of the local and received range. New prekeys will be accepted only if the size

	would stay within this range.
TX Prekeys	A set of one-time-use public key material used to process and initiate CK Creation activities. These public keys are received from the peer entity through CK Prekey activities. Each public prekey has the information defined in Table 7 and is scoped within a single primary SA.

Table 3: Primary Security Association Columns

The information in Table 4 is implicitly scoped to a single primary SA and does not have an explicit relationship in that table. An entity SHOULD remove any TX Content Key when the wall-clock time advances past the end of its validity time interval.

Name	Description
Key ID	A byte string identifier for this key, chosen by the local entity.
Validity Time Interval	An absolute interval of time during which the key is valid for use. This interval is chosen by the local entity.
Peer Accepted	A boolean indicator of whether or not the peer entity has accepted this key. Acceptance is indicated by by receiving step 1 of an associated Key Creation (KC) activity without error.
Partial IV Counter	An unsigned 64-bit integer counter used to construct Partial IV byte strings for outgoing SAFE PDUs as defined in Section 9.1.2. The initial value of each counter for a new content key SHOULD be zero.
Symmetric Key	The actual private key material and intrinsic metadata such as acceptable algorithm types. This logically corresponds to COSE Key information for a symmetric key with structure defined in Section 8.4.

Table 4: TX Content Keys Columns

The information in Table 5 is implicitly scoped to a single primary SA and does not have an explicit relationship in that table. An entity SHOULD remove any RX Content Key when the wall-clock time advances past the end of its validity time interval.

Name	Description
Key ID	A byte string identifier for this key, chosen by the peer entity.
Validity Time Interval	An absolute interval of time during which the key is valid for use. This interval is chosen by the peer entity.
Symmetric Key	The actual private key material and intrinsic metadata such as acceptable algorithm types. This logically corresponds to COSE Key information for a symmetric key with structure defined in Section 8.4.

Table 5: RX Content Keys Columns

The information in Table 6 is implicitly scoped to a single primary SA and does not have an explicit relationship in that table. An entity SHOULD remove any RX Prekey when the wall-clock time advances past the end of its validity time interval.

Name	Description
Key ID	A locally unique byte string identifier for this key.
Validity Time Interval	An absolute interval of time during which the key is valid for use. This interval is chosen by the local entity.
Peer Accepted	A boolean indicator of whether or not the peer entity has accepted this key. Acceptance is indicated by by receiving step 1 of an associated Prekey Creation (PC) activity without error.
Private Key	The actual private key material and intrinsic metadata such as acceptable algorithm types. This logically corresponds to COSE Key information for a private key.

Table 6: RX Prekeys Columns

The information in Table 7 is implicitly scoped to a single primary SA and does not have an explicit relationship in that table. An entity SHOULD remove any TX Prekey when the wall-clock time advances past the end of its validity time interval.

Name	Description
Peer Key ID	A unique byte string identifier for this key, chosen by the peer entity.
Validity Time Interval	An absolute interval of time during which the key is valid for use. This interval is chosen by the peer entity.
Public Key	The actual public key material and intrinsic metadata such as acceptable algorithm types. This logically corresponds to COSE Key information for a public key.

Table 7: TX Prekeys 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 8.

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 6.10.
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 6.12.
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 6.12.
Security Operation Selector	This is a combination of information derived from the negotiated Security Operation Selectors (SOS) during the Section 5.4 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 Options	This field represents a set of context-specific 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 6.14 and per-context specifications (see Section 9).
Secondary PRK	Internal key material derived from the EDHOC session for this SA, as the byte string PRK_SA defined in Section 8.1. This is used to derive further values for content key material as defined in Section 8.5.
TX Content Key Depth Range:	The agreed range of valid size of the TX Content Key store for this SA, which is the intersection of the local and received range. New content keys will be generated as needed to stay within this range.
RX Content Key Depth Range:	The agreed range of valid size of the RX Content Key store for this SA, which is the intersection of the local and received range. New content keys will be accepted only if the size would stay within this range.
TX and RX Content Keys	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 8.5 and per-context specifications (see Section 9).
RX Prekey Depth Range:	The agreed range of valid size of the RX Prekey store for this SA, which is the intersection of the local and received range. New prekeys will be generated as needed to stay within this range.
RX Prekeys	A set of one-time-use private asymmetric key material used to process received CK Creation activities. These private keys are generated locally and their associated public prekeys are communicated to the peer entity via CK Prekey activities. Each private prekey has the information defined in Table 6 and is scoped within a single secondary SA.
TX Prekey Depth Range:	The agreed range of valid size of the TX Prekey store for this SA, which is the intersection of the local and received range.

	New prekeys will be accepted only if the size would stay within this range.	
TX Prekeys	A set of one-time-use public key material used to process and initiate CK Creation activities. These public keys are received from the peer entity through CK Prekey activities. Each public prekey has the information defined in Table 7 and is scoped within a single secondary SA.	

Table 8: 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 necessary metadata, into a PDU for transport between SAFE entities. One of the metadata items is a partial IV (PIV) value used for SAFE ciphertext; another is a security association identifier (SAI) used by the receiver to correlate decoding state.

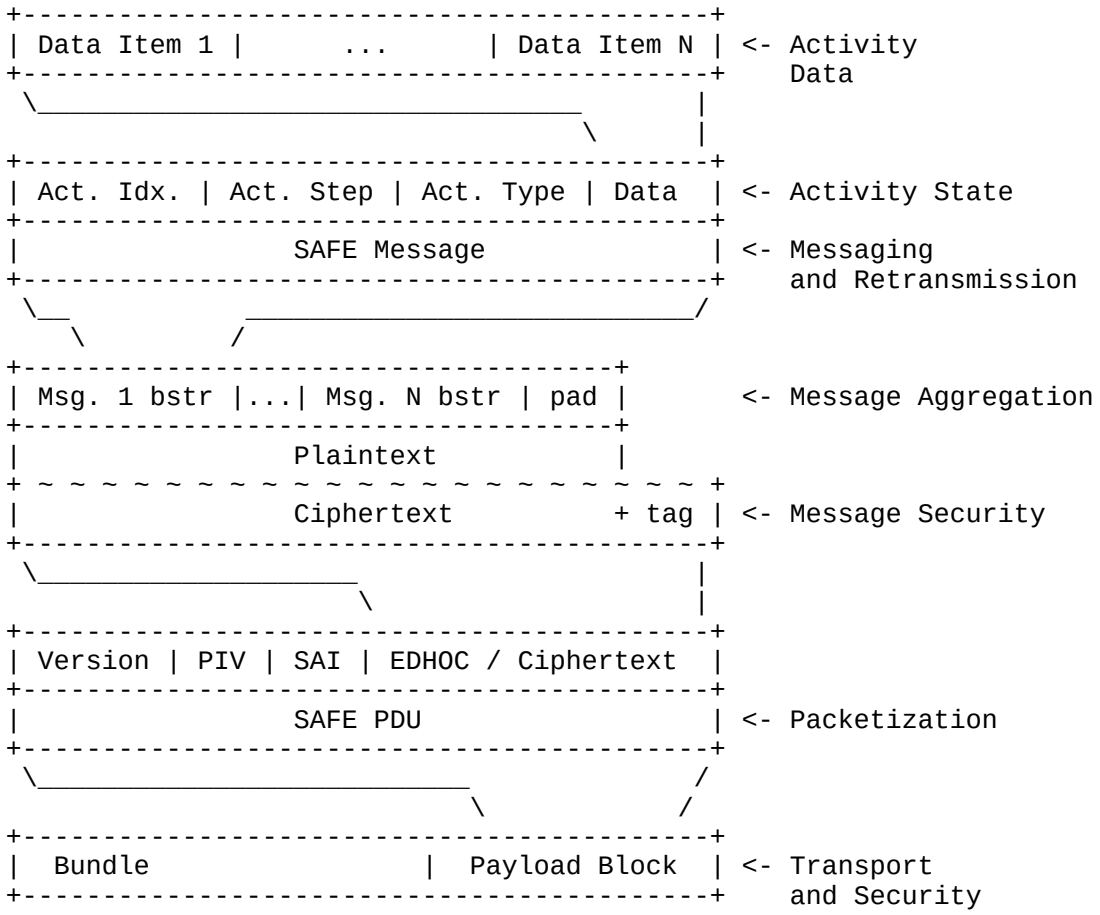


Figure 8: Breakdown of sub-layers within SAFE

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

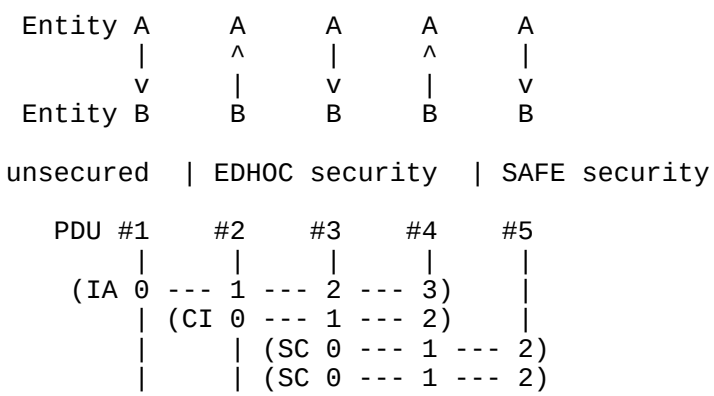


Figure 9: Pipelining of Activities

As described in Section 4.4 and indicated in Figure 9, the IA step-0 message is always sent alone in a PDU and unsecured. After that IA step 1 through 3 messages use EDHOC session encryption to provide confidentiality with EDHOC EAD for SAFE message aggregation. All subsequent messaging occurs using SAFE ciphertext confidentiality with SAFE plaintext aggregation.

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.

Payload Data or Error Indication: This item is either a payload data (Section 4.2.1), which is a CBOR map used to contain data items specific to the logical step in the activity from the containing message, or an error indication (Section 4.2.2), which is a signed integer.

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 an entity SHALL increment the activity index by one. A consequence of this is that although SAFE is delay-tolerant and loss-tolerant, it does operate with the state of the two entities in lockstep as activities are initiated by each peer.

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 first step of an activity SHALL NOT contain an error indication, as there are no previous steps or data items to have caused an error and each activity has a independent state progression (see Section 4.3). Any messages after the first step of an activity SHALL

contain either a payload or an error indication. The determination of if and when to send an error indication is a determination of the sending entity.

The last message of an activity sequence SHALL NOT contain a payload; this message is purely to acknowledge the receipt of the previous step and conclude the activity. The last message MAY contain an error indication, acknowledging the preceding step but indicating the activity has failed.

This is indicated by the following CDDL for the general SAFE message structure and a generic for defining new activity types.

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

safe-msg = $safe-msg .within safe-msg-struct
safe-msg-struct = [
    act-idx: uint,
    act-step: uint,
    act-type: int16,
    ? (safe-map / safe-error)
]
; Signed integer that fits in 16-bit two’s complement form
int16 = -32768 .. 32767

; Generic for defining activity types and their payload
safe-act-type<act-type, act-pyld> = [
    act-idx: uint,
    act-step: uint,
    act-type: act-type,
    ? (act-pyld / safe-error)
]
```

4.2.1. Payload Data Map

All activity-type-specific data items are encoded as a CBOR map having integer keys limited to the inclusive range -32768 to 32767. The interpretation of each item (label and data) of the payload map depends on the activity type of the parent SAFE message, as defined in Section 5 and registered with IANA in Table 33.

This is indicated by the following CDDL, which is also used for generic data maps within SAFE messages.

```
safe-map = {
    * safe-label => safe-value,
}
; Generic map label
safe-label = int16
; Generic map value
safe-value = any
```

4.2.2. Error Indication

Rather than containing a payload, a safe message can instead contain an error indication used by its sender to signal a failure to process a received message or otherwise make progress in an activity. The error indication SHALL consist of an error type value as 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 with IANA in Table 34. Private and experimental use error values are negative and not registered.

```
safe-error = int16
```

4.3. Activity 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 of the message 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 10. 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. In that diagram, if a state transition has a specific trigger condition it is indicated by square bracket text.

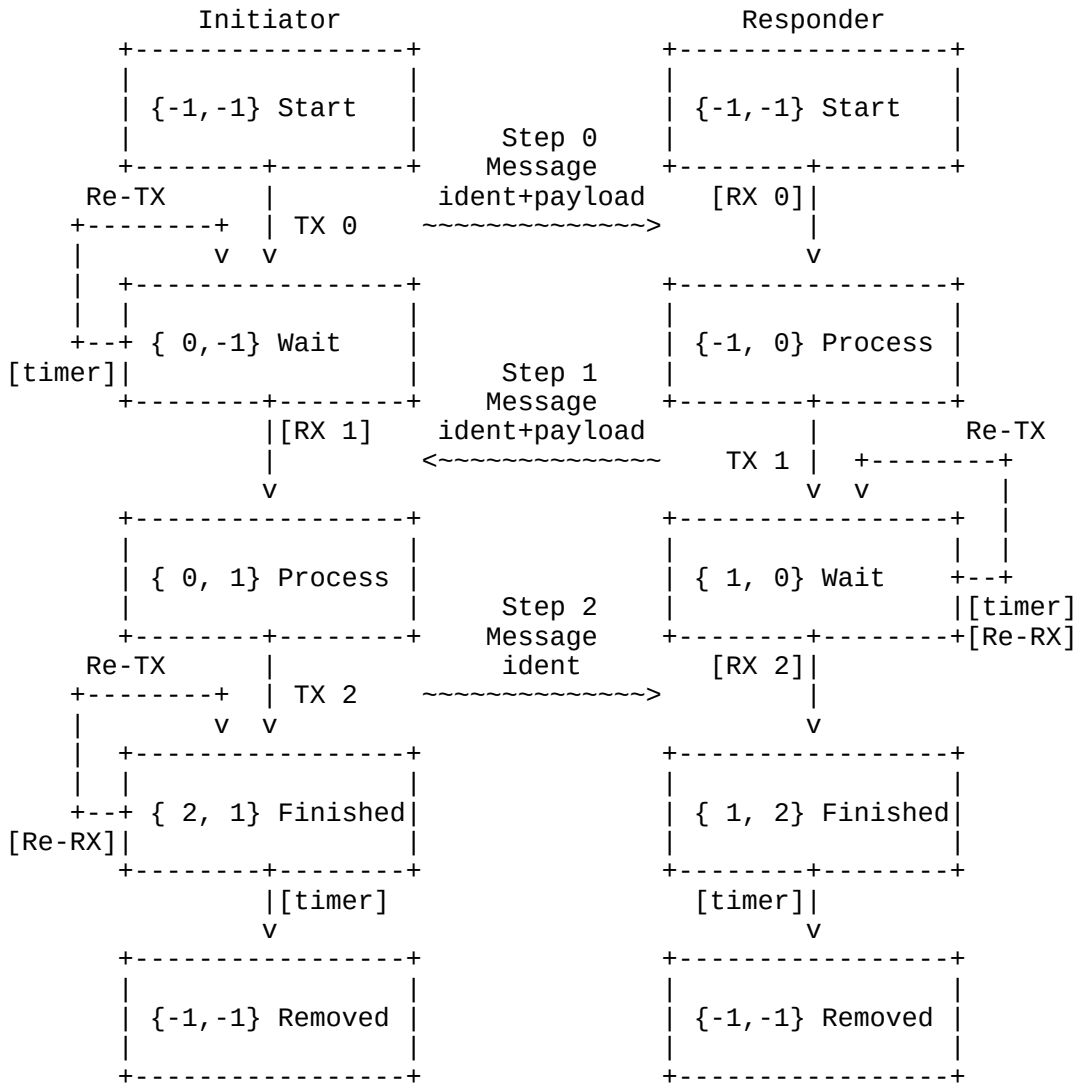


Figure 10: Example activity states for a two-step activity

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 activity-associated retransmission timer expires, the entity SHALL perform the following:

1. Compute the next step number as either: one more than the LRX step, if it is valid, or step 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 associated 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.

Failure of an activity can be accomplished by terminating an activity early using an error indication instead of payload data. When an error is indicated on an activity it SHALL be considered finished regardless of the step in which the error was indicated. Retransmission of a message containing an error indication occurs exactly as any other SAFE message. The logic for if and when to retry an activity which has failed with an error indication is implementation matter and will likely depend on the type of activity which has failed. Any subsequent retry SHALL take the form of a new activity of the same type, as the earlier activity has finished. The retry can have its options adjusted based on the specific error type encountered on the earlier activity.

Rather than indicating an error, a softer form of failure can be handled using conditions within data item(s) specific to the activity type. This is why some activities include a mechanism for providing multiple proposals during initiation and allowing the peer to choose a more narrow condition in its response.

4.4. Message Aggregation

The message aggregation sub-layer allows one or more encoded SAFE messages (as opaque byte strings) to be concatenated together, along with optional padding, into a single plaintext as defined in Figure 11. The entire SAFE plaintext SHALL be a CBOR sequence. Each SAFE message SHALL be represented as a CBOR byte string item containing a single encoded SAFE message as defined in Section 4.2.

If present, the padding SHALL be the last item in the plaintext sequence. The padding SHALL be identified by CBOR tag 55799, indicating "Self-described CBOR" as defined in Section 3.4.6 of [RFC8949]. This tag is arbitrary and provides no additional semantics, it is solely used to distinguish between SAFE message byte strings in the plaintext. It is an implementation matter to determine when and how much padding is added to any SAFE plaintext. A receiver of SAFE plaintext SHALL ignore any padding.

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

; TODO this needs to be redefined and used as part of SAFE confidentiality
safe-pdu-aad = (
  rx-sai: safe-sai
)
```

Figure 11: SAFE plaintext structure CDDL

4.5. Packetization

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 IV byte string or the null value to indicate EDHOC security.

Key Identifier: A key identifier within the primary SA or the null value to indicate EDHOC security.

Receiver SAI: A receiver SAI byte string or integer (based on EDHOC compressed byte string encoding) or the true value.

When the Partial IV and Key Identifier are both null, the payload SHALL be one of the EDHOC messages defined in [RFC9528] and handled in accordance with Section 5.1 and Section 9.1.1. 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 and the Key Identifier are not null, 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.

Any other combinations of header values SHALL be treated as invalid and discarded by a receiver.

These cases are indicated by the following CDDL for the SAFE bundle PDU sequence.

```
; Encoded to PDU as CBOR sequence without array head
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,
    psa-kid: null,
    rx-sai: true,
    ; Group message_1 from RFC 9528
    message_1
)
safe-pdu-edhoc //= (
    partial-iv: null,
    psa-kid: 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 as critical EAD items
edhoc_234 = bstr

safe-pdu-confidential //= (
    partial-iv: bstr,
    psa-kid: safe-kid,
    rx-sai: safe-sai,
    ; Ciphertext data corresponding to ‘safe-pdu-plaintext‘ rule
    ciphertext: bstr
)
```

Figure 12: SAFE PDU structure CDDL

Within restrictions defined for each message type, multiple messages MAY be combined into a single PDU, as either EDHOC EAD (Section 5.1) or SAFE plaintext (Section 4.4). Due to the logic of the IA activity sequencing, only one EDHOC session can make progress between two endpoints at any time so there is no concept of a full EDHOC PDU embedded within an EAD item, only individual messages.

4.6. PDU Transport

Each SAFE PDU is handled as 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 SAFE bundles SHALL NOT be a null EID. Each endpoint for SAFE messaging needs to be an identified singleton. The bundle source and destination EID for received bundles SHALL be exposed to the SAFE entity 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 entities 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 SAFE entity to allow it to de-duplicate and order received SAFE 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.

Each activity type definition SHALL include how many steps comprise a single sequence of messages and what the required and optional payloads are for 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 IA activity does not follow the same messaging structure, it does have an activity identifier allocated to it in Table 32 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 IA 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.

An indication of failure in the activity is provided by an EDHOC error CBOR sequence in place of the normal EDHOC message. SAFE entities SHALL use EDHOC errors in accordance with Section 6 of [RFC9528].

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 entities.

The CDDL corresponding to this activity type are the first two variations of safe-pdu-seq from Section 4.5. Additionally, the _EAD_ payload of the EDHOC messages are extended to be able to confidentially carry encoded SAFE messages during the IA activity. Any number of EAD items of label -
// TBA5 and value matching the safe-msg-bstr rule (defined in Section 4.2) MAY be present in EDHOC lists EAD_2, EAD_3, and EAD_4. EAD items of label
// TBA5 SHALL NOT be present in EAD_1, which is transported as plaintext. When present, all SAFE EAD items SHALL use the negative label to indicate that the item is critical and cannot be ignored by the receiver.

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. Each entity SHALL populate the primary SA with one TX Content Key and one RX Content Key each with a KID of the empty byte string as defined in Section 8.4.
// what about validity time of CKs? Any time after the first reception or transmission of IA step 2, either entity MAY initiate other activities which refer to the new SA. Before that step has occurred the primary SA is still being negotiated and has not yet been created.

5.1.1. EDHOC Requirements

After receiving the ID_CRED_x values from a peer, each SAFE entity SHALL validate the contained or referenced credential in its participating peers (Section 3.1) information base for the associated bundle source.

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.

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 1.

If the receiver of CI step 0 or step 1 does not find its content acceptable it SHALL reply with an error indication, not create any associated primary SA, and abort the parent IA activity. If a CI step 1 or 2 contains an error indication, the receiver SHALL not create any associated primary SA, and abort the parent IA activity.

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: Either no payload, indicating pure acknowledgement, or a map containing a single error item.

Label	Type
1	A Concurrent Activity Support (CAS) limit.
2	An EID Scheme Support (ESS) list.
3	A BPSec Context Support (BCS) list.
Options for subordinate key stores	
10	A Key Depth Range (KDR) for RX content keys supported in the primary SA. This range SHALL have a lower limit greater than zero to indicate that there can always be at least one RX content key.
11	A Key Depth Range (KDR) for TX content keys used in the primary SA. This range SHALL have a lower limit greater than zero to indicate that there will always be at least one TX content key.
12	A Key Depth Range (KDR) for RX content prekeys used in the primary SA. This range MAY have an upper limit of zero to indicate FS will never be used by the sending entity.
13	A Key Depth Range (KDR) for TX content prekeys supported in the primary SA. This range MAY have an upper limit of zero to indicate FS will never be allowed by the sending entity.

Table 9: CI Data Items

```
$safe-msg /= safe-act-type<1, ci-data>
ci-data = { * $$ci-data } .within safe-map

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

$$ci-data //= (10: key-depth-range)
$$ci-data //= (11: key-depth-range)
$$ci-data //= (12: key-depth-range)
$$ci-data //= (13: key-depth-range)
```

It is not until reception of a peer's CI information that later activities can be initiated with the expectation that they will succeed. Any time after the first reception of CI step 0 or 1, the receiving entity MAY initiate other activities which rely on the capabilities expressed in the CI data items.

5.3. Event Notification (EN)

This activity is used by the initiator to inform the peer entity of an event which affects the entire primary SA. The Event Notification (EN) 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.

* Step 1: No payload, message is acknowledgement.

Label	Type
1	The event reason code, as an integer interpreted according to the IANA table // TBD.
2	Event cause message, as an array containing the two-item identity of that message.

Table 10: Event Notification Data Items

```
$safe-msg /= safe-act-type<2, en-data>
en-data = {* $$en-data } .within safe-map

; TBD are there built-in events to be notified of outside of activities?
```

5.4. 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
1	The local SAI per Section 6.5 for the SA which does not yet exist
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
Options for subordinate key stores	

10	A Key Depth Range (KDR) for RX content keys supported in the secondary SA. This range MAY have an upper limit of zero to indicate that no RX content keys can be accepted and that associated traffic toward the CI sender cannot have security applied.
11	A Key Depth Range (KDR) for TX content keys used in the secondary SA. This range MAY have an upper limit of zero to indicate that no TX content keys will be created and associated traffic from the CI sender cannot have security applied.
12	A Key Depth Range (KDR) for RX content prekeys used in the secondary SA. This range MAY have an upper limit of zero to indicate FS will never be used by the sending entity.
13	A Key Depth Range (KDR) for TX content prekeys supported in the secondary SA. This range MAY have an upper limit of zero to indicate FS will never be allowed by the sending entity.
14	An Initial Content Key (ICK) item from the sender’s side. Each entity MAY include this item to indicate initial TX content keys created along with the new SA.

Table 11: SA Creation Data Items

```
$safe-msg /= safe-act-type<3, sc-data>
sc-data = {* $$sc-data } .within safe-map

; SAI for the sender
$$sc-data //= (1: safe-sai)

; 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 content keys
$$sc-data //= (4: safe-sos)
$$sc-data //= (5: safe-kus)

$$sc-data //= (10: key-depth-range)
$$sc-data //= (11: key-depth-range)
$$sc-data //= (12: key-depth-range)
$$sc-data //= (13: key-depth-range)
$$sc-data //= (14: safe-ick)
```

Upon the first reception or transmission of SC step 1, the entity SHALL create a secondary SA based on the data derived in Section 8.1 with no initial content keys or prekeys. Any time after the first reception or transmission of SC step 1, either entity MAY initiate other activities which refer to the new SA. Before that step has occurred the secondary SA is still being negotiated and has not yet been created.

5.5. SA Teardown (ST)

This activity is used to negotiate removal of an established SA from both entities. Either peer can initiate this activity for any reason at any time after the SA is created. The SA Teardown (ST) 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

Table 12: SA Teardown Data Items

```
$safe-msg /= safe-act-type<4, st-data>
st-data = {* $$st-data } .within safe-map

; SAI for the sender
$$st-data //= (1: safe-sai / true)
```

Upon the first reception or transmission of ST step 1, the entity SHALL remove the referenced SA from its associated information bases. Any time after the first reception or transmission of ST step 1, either entity SHALL NOT refer to the removed SA from any other activities.

5.6. Key Creation (KC)

This activity is used to establish a new set of derived content key material within an existing SA without changing the other parameters of the SA (_e.g._, algorithm choice or endpoint selectors). The Key Creation (KC) activity SHALL be identified by activity type 6.

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 2: No payload, message is acknowledgement.

Label	Type
1	The initiator local SAI for the containing SA, as defined in Section 6.5
2	The KID for the content key created by the initiator of this activity, as defined in Section 6.6. From the initiator side this is a TX content key, from the responder side this is an RX content key.
10	An optional identifier for a prekey, shared by the responder in an earlier Prekey Creation (PC) activity, as defined in Section 6.6. From the initiator side this is a TX prekey, from the responder side this is an RX prekey.
11	An optional random nonce per Section 6.9 used to add entropy to the new key. The combination of KID and ARN SHOULD contain at least 128 bits of entropy (see BCP 106 [RFC4086]).
12	A Node Time Interval (NTI) used as the content key validity time interval.

Table 13: Key Creation Data Items

```
$safe-msg /= safe-act-type<6, kc-data>
kc-data = {* $$kc-data } .within safe-map

; Identity of the new CK from the initiator
$$kc-data //= (1: safe-sai)
$$kc-data //= (2: safe-kid)

; Creation parameters
$$kc-data //= (10: safe-kid)
$$kc-data //= (11: safe-arn)
$$kc-data //= (12: safe-nti)
```

Upon the first reception or transmission of KC step 0, the entity SHALL correlate the responder SAI of a Primary SA or Secondary SA and create a new content key within it based on the derived secrets defined in Section 8.4 or Section 8.5 respectively. Any time after the first reception or transmission of KC step 0, either entity MAY initiate Key Discard (KD) or Key Reject (KR) activities which refer to the new content key by its KID.

5.7. Key Discard (KD)

This activity is used to indicate that an earlier content key has been discarded from its TX side owner and will no longer be used for any data plane security. The Key Discard (KD) activity SHALL be identified by activity type 7.

It is an implementation matter to determine when to initiate a Key Discard activity. The Key Discard SHOULD NOT be initiated after the validity time interval of the content key has expired, as it will not serve any purpose to the responder.

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: No payload, message is acknowledgement.

Label	Type
1	The initiator local SAI for the containing SA, as defined in Section 6.5
2	The KID for the content key created by the initiator of this activity, as defined in Section 6.6. From the initiator side this is a TX content key, from the responder side this is an RX content key.

Table 14: Key Discard Data Items

```
$safe-msg /= safe-act-type<7, kd-data>
kd-data = {* $$kd-data } .within safe-map

; Identity of the CK from the initiator
$$kd-data //= (1: safe-sai)
$$kd-data //= (2: safe-kid)
```

Upon the first transmission of KD step 0, the entity SHALL correlate the contained SAI with the local SAI of a Primary SA or Secondary SA and correlate the contained KID with the KID of one of the TX Content Keys and remove that record. Upon the first reception of KD step 0, the entity SHALL correlate the contained SAI with the peer SAI of a Primary SA or Secondary SA and correlate the contained KID with the KID of one of the RX Content Keys and remove that record. Any time

after the first reception or transmission of KD step 0, either entity SHALL NOT refer to the removed content key from any other activities. After transmission of KD step 0, the removed content key SHALL NOT be used for any later SAFE or data plane security.

5.8. Key Reject (KR)

This activity is used to indicate that an earlier content key has been rejected from its RX side and will no longer be used for any data plane security. The Key Reject (KR) activity SHALL be identified by activity type 8.

It is an implementation matter to determine when to initiate a Key Reject activity. The Key Reject SHOULD NOT be initiated after the validity time interval of the content key has expired, as it will not serve any purpose to the responder.

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: No payload, message is acknowledgement.

Label	Type
1	The initiator local SAI for the containing SA, as defined in Section 6.5
2	The KID for the content key created by the initiator of this activity, as defined in Section 6.6. From the initiator side this is an RX content key, from the responder side this is a TX content key.

Table 15: Key Reject Data Items

```
$safe-msg /= safe-act-type<8, kr-data>
kr-data = {* $$kr-data } .within safe-map

; Identity of the CK from the initiator
$$kr-data //= (1: safe-sai)
$$kr-data //= (2: safe-kid)
```

Upon the first transmission of KR step 0, the entity SHALL correlate the contained SAI with the local SAI of a Primary SA or Secondary SA and correlate the contained KID with the KID of one of the RX Content Keys and remove that record. Upon the first reception of KR step 0, the entity SHALL correlate the contained SAI with the peer SAI of a Primary SA or Secondary SA and correlate the contained KID with the KID of one of the TX Content Keys and remove that record. Any time after the first reception or transmission of KR step 0, either entity SHALL NOT refer to the removed content key from any other activities. After reception of KR step 0, the removed content key SHALL NOT be used for any later SAFE or data plane security.

5.9. Prekey Creation (PC)

This activity is used to preemptively share a one-time-use public key which can be used to add forward secrecy to later-derived content keys through the Key Creation (KC) activity. The initiator of this activity will hold a corresponding private key for the lifetime of the shared prekey. The recipient of this activity will be the initiator of the later key creation (Section 5.6) activity. The Prekey Creation (PC) activity SHALL be identified by activity type 9.

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 2: No payload, message is acknowledgement.

Label	Type
1	The initiator local SAI for the containing SA, as defined in Section 6.5
3	A Node Time Interval (NTI) used as the prekey validity time interval.
4	The sender’s one-time-use public prekey, as defined in Section 6.8. This public key SHALL contain a kid (2) parameter used to identify the prekey. This public key SHALL contain an alg (1) parameter matching the "EDHOC key exchange algorithm" from the EDHOC cipher suite.

Table 16: Prekey Creation Data Items

```
$safe-msg /= safe-act-type<9, pc-data>
pc-data = {* $$pc-data } .within safe-map

; Identifiers for the initiator
$$pc-data //= (1: safe-sai)
$$pc-data //= (2: safe-kid)

$$pc-data //= (3: safe-nti)
$$pc-data //= (4: safe-pks)
```

Upon the first reception of PC step 0, the entity SHALL correlate the Local SAI of a Primary SA or Secondary SA and record the prekey information within TX Prekey store of the SA. Any time after the first reception of PC step 0, the entity MAY initiate Key Creation (KC) activities which refer to the new prekey by its KID.

5.10. Prekey Discard (PD)

This activity is used to indicate that an earlier private prekey has been discarded from its RX side owner and the associated public key will no longer be usable by any KC activity. The Prekey Discard (PD) activity SHALL be identified by activity type 10.

It is an implementation matter to determine when to initiate a Prekey Discard activity. The Prekey Discard SHOULD NOT be initiated after the validity time interval of the prekey has expired, as it will not serve any purpose to the responder.

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: No payload, message is acknowledgement.

Label	Type
1	The initiator local SAI for the containing SA, as defined in Section 6.5
2	The KID for the prekey created by the initiator of this activity, as defined in Section 6.6. From the initiator side this is a RX content prekey, from the responder side this is a TX content prekey.

Table 17: Prekey Discard Data Items

```
$safe-msg /= safe-act-type<10, pd-data>
pd-data = {* $$pd-data } .within safe-map

; Identity of the prekey from the initiator
$$pd-data //= (1: safe-sai)
$$pd-data //= (2: safe-kid)
```

Upon the first transmission of PD step 0, the entity SHALL correlate the contained SAI with the local SAI of a Primary SA or Secondary SA and correlate the contained KID with the KID of one of the RX Content Prekeys and remove that record. Upon the first reception of PD step 0, the entity SHALL correlate the contained SAI with the peer SAI of a Primary SA or Secondary SA and correlate the contained KID with the KID of one of the TX Content Prekeys and remove that record. Any time after the first reception or transmission of PD step 0, either entity SHALL NOT refer to the removed prekey from any other activities.

5.11. Prekey Reject (PR)

This activity is used to indicate that an earlier content prekey has been rejected from its RX side and will no longer be used for any data plane security. The Prekey Reject (PR) activity SHALL be identified by activity type 11.

It is an implementation matter to determine when to initiate a Prekey Reject activity. The Prekey Reject SHOULD NOT be initiated after the validity time interval of the prekey has expired, as it will not serve any purpose to the responder.

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: No payload, message is acknowledgement.

Label	Type
1	The initiator local SAI for the containing SA, as defined in Section 6.5
2	The KID for the content key created by the initiator of this activity, as defined in Section 6.6. From the initiator side this is an TX content prekey, from the responder side this is a RX content prekey.

Table 18: Prekey Reject Data Items

```
$safe-msg /= safe-act-type<11, pr-data>
pr-data = {* $$pr-data } .within safe-map

; Identity of the CK from the initiator
$$pr-data //= (1: safe-sai)
$$pr-data //= (2: safe-kid)
```

Upon the first transmission of PR step 0, the entity SHALL correlate the contained SAI with the local SAI of a Primary SA or Secondary SA and correlate the contained KID with the KID of one of the TX Content Prekeys and remove that record. Upon the first reception of PR step 0, the entity SHALL correlate the contained SAI with the peer SAI of a Primary SA or Secondary SA and correlate the contained KID with the KID of one of the RX Content Prekeys and remove that record. Any time after the first reception or transmission of PR step 0, either entity SHALL NOT refer to the removed content key from any other activities.

6. Data Item Types

This section defines the initial set of data items which can be present in the 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 = safe-map / [+ safe-map]
```

6.1. 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
```

6.2. 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
```

6.3. 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
```

6.4. Key Depth Range (KDR)

This data item indicates a range of valid depth of a key store in the sending entity. The value is an array containing two items: a minimum valid depth as a uint and a maximum valid depth as a uint. The minimum depth SHALL be less than or equal to the maximum depth. Specific uses of this data item can further restrict the valid minimum or maximum.

```
key-depth-range = [
  min: uint,
  max: uint
]
```

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 be limited to a length between 0 and 16 bytes inclusive. This length allows a Universally Unique IDentifier (UUID) [RFC9562] to be used as an SAI. 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 .size (0..16)) / (-24..23)
```

6.6. Key Identifier (KID)

The Key Identifier (KID) data item is used by both sides of an SA to uniquely identify a shared content key or prekey within each entity. All KIDs are guaranteed unique within the parent primary SA context, but each entity is able to reuse KID values across different primary SAs. Each content key has a single KID defined by the entity which initiated the creation of the symmetric key, and each content prekey has a single KID defined by the entity which holds the associated private key.

A KID SHALL be treated as an opaque byte string as its internal representation and comparison logic. A KID SHALL be limited to a length between 0 and 16 bytes inclusive. This length allows a UUID [RFC9562] to be used as an KID. A KID SHALL have the same compressed encoding rules as EDHOC connection identifiers as defined in Section 3.3.2 of [RFC9528].

```
safe-kid = (bstr .size (0..16)) / (-24..23)
```

6.7. Initial Content Key (ICK)

This data item allows creating non-FS/PCS content keys on a new SA during its creation, which saves activity sequencing for those keys. The data value SHALL be a CBOR array of items, each being a CBOR map interpreted according to the Key Creation (KC) activity. For this purpose, the SAI SHALL be omitted from the map for each content key as the parent SA Creation activity already has an explicit SAI for the sender.

```
safe-ick = [+ kc-data]
```

The processing of each item of the array SHALL be interpreted identically to step 0 of the KC activity from Section 5.6. Any failure during processing or creation of each content key SHALL NOT affect the processing or creation of any other key from the same data item.

6.8. Public Key Structure (PKS)

This data item indicates a public key from sender to enable forward secrecy during a later CK Create activity. The algorithm and parameters related to key exchange or encapsulation are taken from the EDHOC cipher suite negotiated during initial authentication.

```
safe-pks = COSE_Key  ; from [RFC 9052]
```

6.9. 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 5.4) and CK Creation (Section 5.6) to generate content key 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)
```

6.10. 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 transport (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,
    transport: 2,
)
```

6.11. 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
]
```

6.12. 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]
```

6.13. 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,
)
```

6.14. 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: 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

This section gives higher-level explanations and requirements for multiple activities between pairwise combinations of peers.

7.1. Concurrent Activities

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.

The _pairwise CAS limit_ is a hard upper bound on in-progress activities between pairs of entities, defined as the minimum of the local CAS limit and the CAS limit received from a peer entity. An entity SHALL NOT initiate any activity which would cause the total number of in-progress activities to exceed the pairwise CAS limit with a peer entity. An entity MAY ignore any messages from a peer which would cause the total number of in-progress activities to exceed its own CAS limit.

Each entity needs to leave some margin within the pairwise CAS limit for its peer to initiate new activities. An entity SHALL NOT initiate any activity which would cause the total number of in-progress activities for which it is the initiator to exceed the pairwise CAS limit with a peer entity less some _CAS margin_. An entity SHALL use a CAS margin of at least one. This always enables a peer to initiate one new activity. An entity SHOULD use a CAS margin that is some proportion to the CAS limit itself. An entity SHOULD allow a peer to initiate an activity which would exceed the local CAS margin. This allows entities to choose different CAS margin for their own reasons without negotiating.

Two examples of how this parameter affects activity sequencing is shown in Figure 13 for a large CAS limit and Figure 14 for a constraining limit of two. Note that when constrained, some activities can only begin with specific odd PDU numbers (from IA initiator) or even numbers (from the IA responder) so there are necessarily gaps between activities so less efficient use of the network.

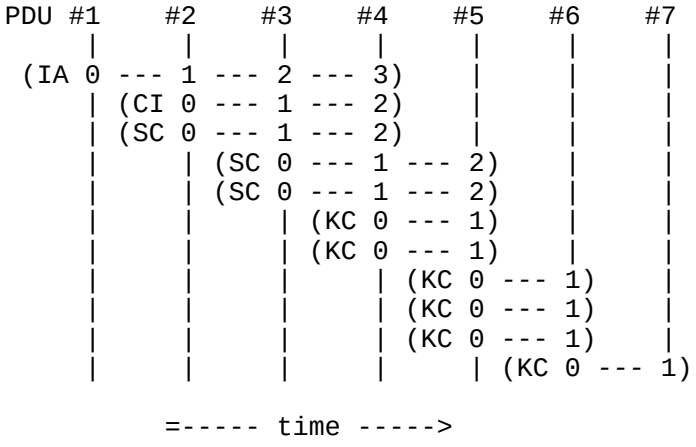
The first example performs the following sequence of activities, using up to 7 concurrent activities:

1. IA to authenticate, initiated by Entity A.

2. Early CI to exchange capabilities of each entity before SAs are established.

3. One secondary SA initiated by Entity B (in PDU #2), with two non-FS CKs created by Entity B (in PDU #4) and two by Entity A (in PDU #5).

4. Two secondary SAs initiated by Entity A (in PDU #3), each with one non-FS CK created by Entity A (in PDU #5) and one by Entity B (in PDU #6).



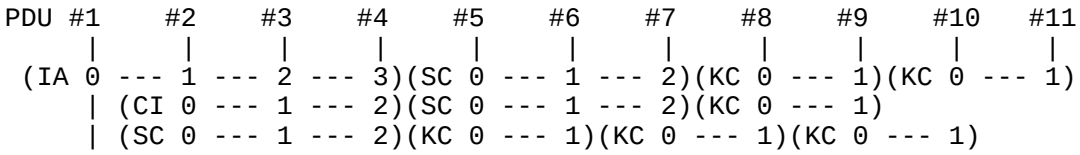
The second example performs the same sequence, limited to 3 concurrent activities (no more than 2 initiated by each side):

1. IA to authenticate, initiated by Entity A.

2. CI to exchange capabilities of each entity before SAs are established.

3. One secondary SA initiated by Entity B (in PDU #2), with two CKs created by Entity B (in PDU #8) and two by Entity A (in PDU #5 and #7).

4. Two secondary SAs initiated by Entity A (in PDU #5), each with one CK created by Entity A (in PDU #9) and one by Entity B (in PDU #10).



7.2. Establishment Ordering

Regardless of how activities are aggregated into specific ADUs, the following requirements apply to ordering of activities related to a single primary SA. These build upon the per-activity constraints defined in Section 5 to define a required or preferred order of activities, specifically during primary SA establishment.

When no primary SA exists between two SAFE application endpoints, the first messaging between those applications SHALL be Initial Authentication (IA) used to establish an initial private channel and mutually authenticate each peer. 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 step 0 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 IA step 1, before the IA initiator has been authenticated, or MAY wait until after the IA activity has completed step 2 and both peers are authenticated. It is an implementation matter to configure whether CI is allowed before mutual authentication.

After step 3 of the IA activity has been received, both peers have authenticated each other and established a primary SA configured with a single content key in each direction, each having indefinite validity time, and no content prekeys. Any additional content keys on the primary SA depend on CC activities by each peer to create those keys.

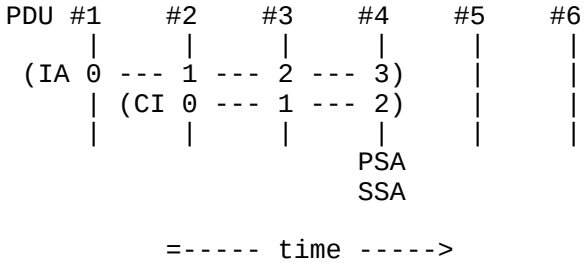


Figure 15: Early CI initiation

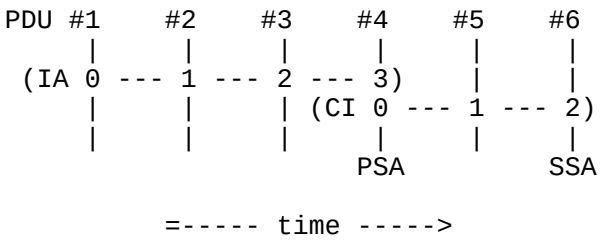


Figure 16: Late CI initiation

Only after step 0 of the CI activity has been received, the responder for the activity SHOULD begin any number of needed Key Creation (KC) activities for the primary SA SA Creation (SC) activities to establish secondary SAs. Only after step 1 of the CI activity has been received, the initiator for the activity SHOULD begin any number of needed Key Creation (KC) activities for the primary SA SA Creation (SC) activities to establish secondary SAs. The number of concurrent activities allowed to be started at each step is limited by the CAS limit for the receiving peer.

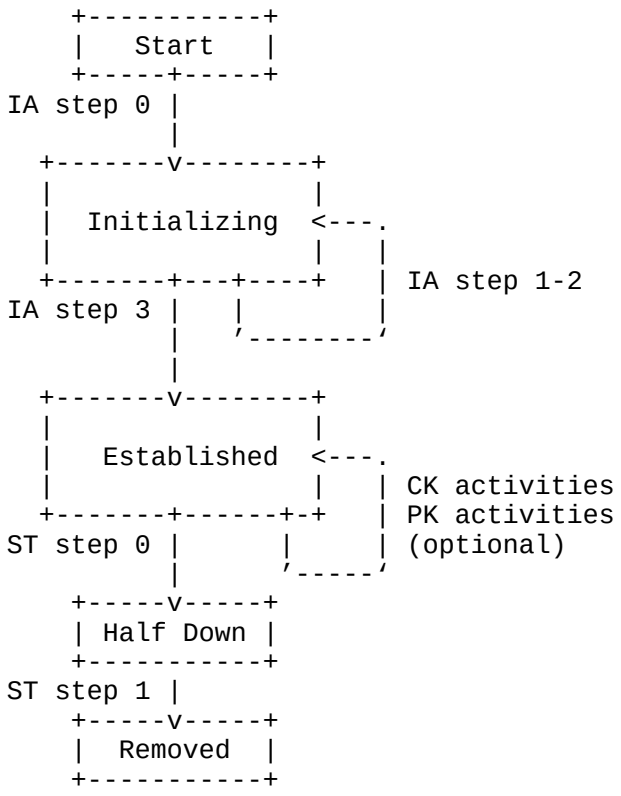


Figure 17: Notional States of a Primary SA

7.3. Content Key Upkeep

Initiating or responding to the creation of a new SA establishes an obligation by each entity to upkeep content keys in that SA. There are two extremes of strategy for keeping content keys replenished in each direction within an SA, described below. Each entity is in control of its own TX content keys and corresponding RX content keys of its peer, and is free to choose a strategy to do so without negotiation. Future extensions to SAFE can define new CI data items and SC data items that allow negotiating such behavior between peers.

Reactive Keying: This strategy is to ensure that one TX content key has a validity time including the current wall-clock time. When that validity time is about to expire perform a CC activity to create a new one. This strategy uses fewer key store entries but requires a just-in-time interactivity to create new keys as needed.

Pre-planned Keying: This strategy is to establish a backlog of current and future TX content keys, either out to a specific horizon of time or a specific depth of number. As the current keys expire, the backlog is added to to maintain the desired horizon or depth. This strategy avoids just-in-time behavior but requires more key store size.

7.4. Prekey Upkeep

Similar to the discussion in Section 7.3 entities which agree to use forward secrecy on some (or all) of their RX content keys need to upkeep RX content prekeys in that SA. Part of that is generating private prekey material locally and the other part is sending public prekey material to the peer entity.

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 as depicted in Figure 18 and defined in the following subsections.

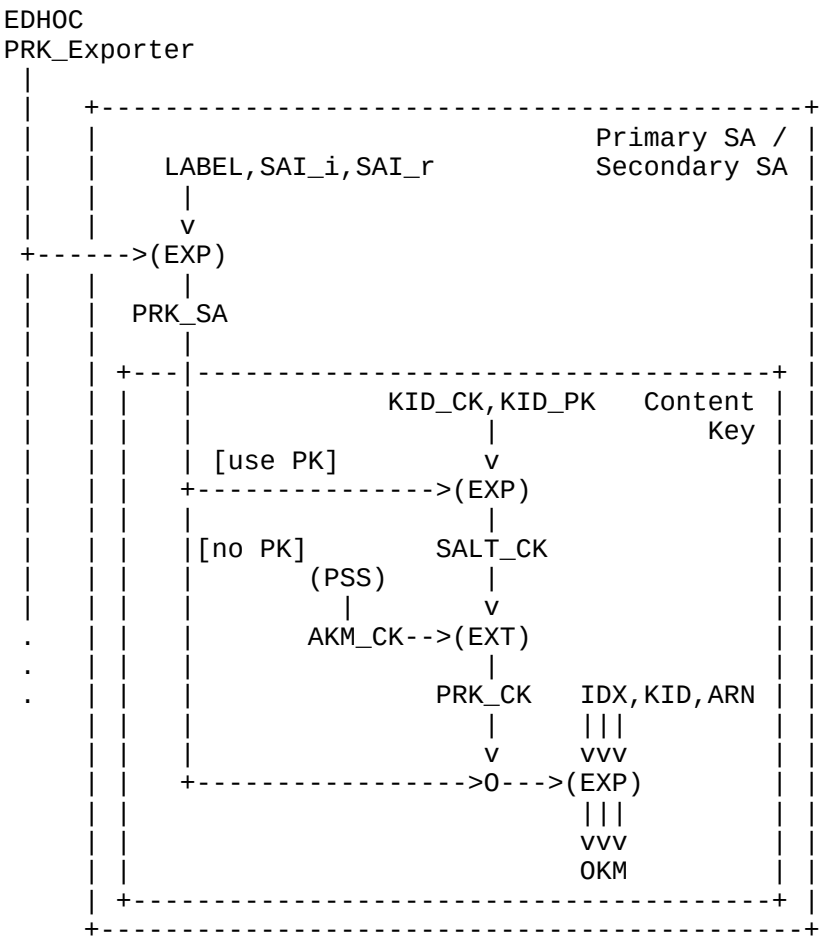


Figure 18: SAFE Key Schedule

An application-level extraction function SAFE_EXT is defined with the same logic as the EDHOC_Extract from Section 4.1.1 of [RFC9528] except using the Hash Algorithm of the Primary SA (which is the "application hash algorithm" from the EDHOC cipher suite). The pseudocode defining this function is in Figure 19.

```
SAFE_EXT(salt, IKM)
  = EDHOC_Extract(salt, IKM)
```

Figure 19: SAFE Extract function

An application-level expansion function SAFE_EXP is defined with the same logic as the EDHOC_Expand in Section 4.1.2 of [RFC9528] except using the Hash Algorithm of the Primary SA (which is the "application hash algorithm" from the EDHOC cipher suite). The pseudocode defining this KDF is in Figure 20.

```
SAFE_EXP(PRK, context, length)
  = EDHOC_Expand(PRK, context | E(length), length)
```

Figure 20: SAFE Expand function

In this document the notation "E()" is used to indicate the deterministically encoded CBOR and the notation "|" is used to indicate the concatenation of encoded items as a CBOR sequence without byte string wrapping. So in the definition of expand function above, the length is encoded and concatenated to the context sequence, which is treated as opaque input bytes to the expand function. This is not a different mechanism than the EDHOC "info" rule sequence, just a different notation in pseudocode.

8.1. Security Association PRK

Each primary and secondary SA is created with a unique SAI from each peer, so the pair of SAI values provides a unique context for deriving an SA-specific, stable PRK of fixed-length (based on the EDHOC cipher suite).

The EDHOC processes which generate the input PRK_Exporter causes that PRK to be bound to all of the transcript hash (TH) data from the EDHOC session, including the ephemeral keys used to establish that session. That EDHOC_Exporter is used once during each SA creation to derive a seed PRK for the entire SA. The PRK of each SA is then used to deterministically derive content keys under that specific SA.

Pseudocode used to explain this is shown in Figure 21, where the named parameters are:

The value
// TBA4: The exporter label allocated to BP SAFE.

SAI_i and SAI_r: The Security Association Identifier (SAI) from the initiator and responder respectively

hash_length: The length of output size of the Hash Algorithm of the Primary SA.

```
PRK_SA = EDHOC_Exporter(TBA4, E(SAI_i) | E(SAI_r), hash_length)
```

Figure 21: Primary SA Creation

8.2. Prekey Shared Secret

The prekey shared secret (PSS) process is used to produce additional key material (AKM) data to enable forward secrecy of derived values. The AKM is not assumed to be uniformly random, so it needs to use a KDF extraction function (with appropriate salt) to derive any PRK values. This is the exact same strategy and mechanism used by EDHOC itself for static DH authentication in Section 4.1.1 of [RFC9528].

When the prekey is an ECDH derivation key, the AKM_CK is the secret G_XY from the ECDH process. When the prekey is a KEM decapsulation key, the AKM_CK is the secret K from the KEM decapsulation process.

8.3. Content Key PRK

Each content key material is derived from a PRK which can either have forward secrecy or not, depending on a choice made by the initiator of the CK Create activity. Either the CK Create activity references an associated Prekey known to the responder, which is used to derive or decapsulate ephemeral additional key material (AKM) used for forward secrecy, or it does not reference a prekey and no forward secrecy is provided.

When no prekey is referenced during SA creation, the PRK_CK used is the associated PRK_SA value of the parent SA.

When a prekey is referenced during SA creation, the PRK_CK is extracted based on a chosen prekey as shown in Figure 22, where the named parameters are:

PRK_SA: The parent SA PRK value.

KID_CK: The KID of the to-be-created content key, unique within the parent SA key store, taken from a Key Identifier (KID) from the initiator.

KID_PK: The KID of the referenced prekey, unique within the parent SA key store, taken from a Key Identifier (KID) from the initiator.

hash_length: The length of output size of the Hash Algorithm of the Primary SA.

AKM_CK: The prekey shared secret, as defined in Section 8.2

```
SALT_CK = SAFE_EXP(PRK_SA, E(KID_CK) | E(KID_PK), hash_length)
PRK_CK = SAFE_EXT(SALT_CK, AKM_CK)
```

Figure 22: Content Key PRK pseudocode

8.4. Primary SA Content Keys

The primary SA content keys 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 creating a new primary SA content key, the COSE key SHALL contain the following parameters.

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

kid (2):
: The KID byte string from the content key information, which may be the empty string.

alg (3):
: The AEAD Algorithm of the primary SA.

key_ops (4):
: Either _encrypt_ (3) if this is a TX Content Key, or _decrypt_ (4) if this is an RX Content Key

Base IV (5):
: The BIV_SA1 value from Figure 23.

k (-1):
: The SK_SA1 value from Figure 23.

Pseudocode used to explain this is shown in Figure 23, where the named parameters are:

TXI
: A boolean value indicating whether the initiator of the content key was the initiator for the parent SA, as CBOR true or false. This ensures that each key (TX or RX) store contains a distinct KID space.

KID_CK:
: The Key Identifier (KID) from the initiator as an encoded CBOR byte string.

ARN:
: The Additional Random Nonce (ARN) from the initiator, or the

empty byte string if no such data item was received, as an encoded CBOR byte string.

PRK_CK: Either the PRK extracted based on the local prekey, if a prekey was referenced by the initiator, or the PRK from the Primary SA.

key_length: The length of the encryption key for the AEAD Algorithm of the Primary SA.

iv_length: The length of the initialization vector for the AEAD Algorithm of the Primary SA.

```
context_ck1 = E(TXI) | E(KID_CK) | E(ARN)

SK_SA1   = SAFE_EXP(PRK_CK, E(1) | context_ck1, key_length)
BIV_SA1 = SAFE_EXP(PRK_CK, E(2) | context_ck1, iv_length)
```

Figure 23: SA Creation pseudocode

8.5. Secondary SA Content Keys

Additional output key material (OKM) specific to the BPSec security context being used is derived from the PRK_SA2 value with integer labels for different, security-context-specific 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 24, 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_EXP(PRK_SA2, ctxid | context, length)
```

Figure 24: Secondary SA derived OKM

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. Select a single key from the TX Content Keys of the Primary SA as defined in Section 3.3. The choice of key is implementation dependent, but SHALL have a validity time interval which includes the current wall-clock time.

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

3. Increment the Partial IV counter associated with the TX content key.

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

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

   Protected Header:
   An encoded map containing the following pairs:

   alg (1):
   The AEAD Algorithm of the Primary SA.

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

   Unprotected Header:
   A map containing the following pairs:

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

   key:
   The COSE key of the selected TX content key from the Primary SA, 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 12, and using the same encoding as the PDU)
// No binding to transport source or destination EID?? Should
// concatenate these in AAD.

6. Use the result to construct the following PDU fields, named as in Section 4.5:

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 and psa-kid values, 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 the PDU is considered to be invalid and is dropped.

2. Use the provided psa-kid to match with one of the RX Content Keys of the Primary SA. If no match is found then the entity MAY retain the PDU for an implementation defined period of time awaiting an associated CK Create activity. Otherwise the PDU is considered to be invalid and is dropped.

3. If the Bundle Creation Time associated with the PDU is not within the validity time interval of the RX Content Key, the PDU is considered to be invalid and is dropped.

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

Protected Header:
An encoded map containing the following pairs:

alg (1):
The AEAD Algorithm of the Primary SA.

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

Unprotected Header:
A map containing the following pairs:

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

key:
The COSE key from the matched RX Content Key (defined in Table 5).

ciphertext:
The ciphertext field of the PDU payload.

external_aad:
A byte string encoding the CBOR Sequence safe-pdu-aad (as defined in Figure 12, and using the same encoding as the PDU).

If decryption fails then the PDU is considered to be invalid and is dropped.

5. Iterate through, decode, and process the CBOR sequence of byte strings items for all messages in the plaintext. The failure to process any message in the plaintext SHALL NOT affect the decoding and processing of any other message in the same plaintext. Any optional padding item in the plaintext is ignored.

For both encryption and decryption, the Base IV (defined in Section 8.4) present in the COSE key of each TX and RX Content Key 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 transport (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 19

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 20

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 25) 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 25: 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 21

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 22

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 26) 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 26: 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 23

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 24

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 25: 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 27) 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 27) if the key is for traffic from the initiator side, or CTX3_BIV_RI otherwise

k (-1):
Either CTX3_KEY_IR (from Figure 27) 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 27: 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 security partial IV (Section 4.5) 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 26: '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 27: 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	A single message for BPv7 SAFE	[This specification]

Table 28: 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 29 and initial contents of Table 30.

Range	Registration Procedures
-32768 to 0	Reserved
1 to 255	RFC Required
256 to 32767	Specification Required

Table 29: 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	transport	[This specification]
3 to 32767	_Unassigned_	

Table 30: SAFE Security Modes

Within the registry group of [IANA-SAFE], the registry titled "SAFE Activity Types" has been created with registration procedures from Table 31 and initial contents of Table 32.

Range	Registration Procedures
-32768 to 0	Reserved
1 to 255	RFC Required
256 to 32767	Specification Required

Table 31: 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	Event Notification	EN	Section 5.3 of [This specification]
3	SA Creation	SC	Section 5.4 of [This specification]
4	SA Teardown	ST	Section 5.5 of [This specification]
6	Key Creation	SC	Section 5.6 of [This specification]
7	Key Discard	KD	Section 5.7 of [This specification]
8	Key Reject	KR	Section 5.8 of [This specification]
9	Prekey Creation	PC	Section 5.9 of [This specification]
10	Prekey Discard	PD	Section 5.10 of [This specification]
11	Prekey Reject	PR	Section 5.11 of [This specification]
12 to 32767	_Unassigned_		

Table 32: 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]
IA		_this activity does not use data item labels_		
CI	1	Concurrent Activity Support	CAS	Section 6.1 of [This specification]
CI	2	EID Scheme Support	ESS	Section 6.2 of [This specification]
CI	3	BPSec Context Support	BCS	Section 6.3 of [This specification]
CI	10	RX content key depth range	KDR	Section 6.4 of [This specification]
CI	11	TX content key depth range	KDR	Section 6.4 of [This specification]
CI	12	RX content prekey depth	KDR	Section 6.4 of [This

		range		specification]
CI	13	TX content prekey depth range	KDR	Section 6.4 of [This specification]
SC	1	Security Association Identifier	SAI	Section 6.5 of [This specification]
SC	4	Security Operation Selector	SOS	Section 6.13 of [This specification]
SC	5	Key Use Selector	KUS	Section 6.14 of [This specification]
SC	6	Endpoint Selector at Initiator	ESI	Section 6.12 of [This specification]
SC	7	Endpoint Selector at Responder	ESR	Section 6.12 of [This specification]
SC	8	Validity Node Time Interval	NTI	Section 6.11 of [This specification]
SC	9	Security Mode Selector	SMS	Section 6.10 of [This specification]
SC	10	RX content key depth range	KDR	Section 6.4 of [This specification]
SC	11	TX content key depth range	KDR	Section 6.4 of [This specification]
SC	12	RX content prekey depth range	KDR	Section 6.4 of [This specification]
SC	13	TX content prekey depth	KDR	Section 6.4 of [This

		range		specification]
ST	1	Security Association Identifier	SAI	Section 6.5 of [This specification]
KC	1	Security Association Identifier	SAI	Section 6.5 of [This specification]
KC	2	Key Identifier	KID	Section 6.6 of [This specification]
KC	10	Prekey Identifier	KID	Section 6.6 of [This specification]
KC	11	Additional Random Nonce	ARN	Section 6.9 of [This specification]
KC	12	Validity Node Time Interval	NTI	Section 6.11 of [This specification]
KD	1	Security Association Identifier	SAI	Section 6.5 of [This specification]
KD	2	Key Identifier	KID	Section 6.6 of [This specification]
KR	1	Security Association Identifier	SAI	Section 6.5 of [This specification]
KR	2	Key Identifier	KID	Section 6.6 of [This specification]
PC	1	Security Association Identifier	SAI	Section 6.5 of [This specification]
PC	2	Prekey Identifier	KID	Section 6.6 of [This

				specification]
PC	3	Validity Node Time Interval	NTI	Section 6.11 of [This specification]
PC	4	Public Key Structure	PCS	Section 6.8 of [This specification]
PD	1	Security Association Identifier	SAI	Section 6.5 of [This specification]
PD	2	Key Identifier	KID	Section 6.6 of [This specification]
PR	1	Security Association Identifier	SAI	Section 6.5 of [This specification]
PR	2	Key Identifier	KID	Section 6.6 of [This specification]

Table 33: 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	0	Invalid activity index	[This specification]

all types	1	Unexpected activity step	[This specification]
all types	2	Unknown activity type	[This specification]
all types	3	Unexpected payload	[This specification]
all types	10	Invalid data item label	[This specification]
all types	11	Unknown data item label	[This specification]
all types	12	Invalid data item value	[This specification]
SC	13	Duplicate SAI	[This specification]
ST,KC,KD,KR,PC,PD,PR	14	Unknown security association	[This specification]
KC,PC	15	Duplicate KID	[This specification]
KD,KR	16	Unknown content key	[This specification]
KC,PD,PR	17	Unknown content prekey	[This specification]

Table 34: 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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[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-17, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-17>.

[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-15, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-dtn-bpsec-cose-15>.

[I-D.ietf-dtn-eid-pattern]
Sipos, B., "Bundle Protocol Endpoint ID Patterns", Work in Progress, Internet-Draft, draft-ietf-dtn-eid-pattern-07, 16 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-dtn-eid-pattern-07>.

[I-D.pocero-authkem-edhoc]
Fraile, L. P., Koulamas, C., Fournaris, A. P., and E. Haleplidis, "KEM-based Authentication for EDHOC", Work in Progress, Internet-Draft, draft-pocero-authkem-edhoc-02, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-pocero-authkem-edhoc-02>.

[I-D.spm-lake-pqsuites]
Selander, G. and J. P. Mattsson, "Quantum-Resistant Cipher Suites for EDHOC", Work in Progress, Internet-Draft, draft-spm-lake-pqsuites-01, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-spm-lake-pqsuites-01>.

13.2. Informative References

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

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

[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>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/info/rfc9562>.

[github-dtn-bp-safe] Sipos, B., "Bundle Protocol (BP) Security Associations with Few Exchanges (SAFE)", <https://github.com/BrianSipos/dtn-bp-safe/>.

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

Appendix A. Example Sequencing

This section contains a minimal example of two nodes initializing a primary SA and one secondary SA 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.

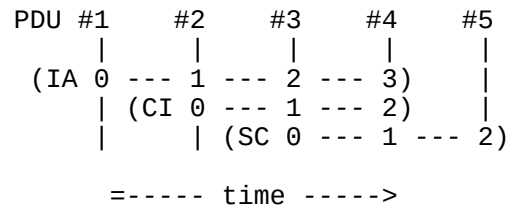


Figure 28: Activities within PDUs in this example

A.1. SAFE PDU #1

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

```
01f6f6f50006582031f82c7b5b9cbbf0f194d913cc12ef1532d328ef32632a4881a1
c0701e237f042d
```

The contents of that PDU are following CBOR sequence:

```
1, / version /
null, / partial-iv /
null, / psa-kid /
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:

```
01f6f62d5883dc88d2d51da5ed67fc4616356bc8ca74ef9ebe8b387e623a360ba480
b9b29d1c26c7c32e544dbc92288007d943689e6d03994b318a63da154aaa242318bd
82713c3e0f3468febdebe25d13723ea6553bb48f7005c1eaba68c83550c86cdc69b2
26fd064affca66c35a1e58bc635133f9f248fed15e42baceb0d0ca8727dc1b612f92
e1
```

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

```
1, / version /
null, / partial-iv /
null, / psa-kid /
-14 / rx-sai from C_I /
/ message_2: h'dc88...92e1' /
```

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'7934E54041AE5ACB7C68DE7ED477D947E4214E84659D99FCD49C4B25033DD92AF5
0DE0570BC27B72039CEB2F4A4F8F05082FF9739A823C25634FE42BB306F296',
-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:

```
01f6f6411858b3e418256972ed10356c2cadc3e72dd86cddf060b8b0b4e2582de839
c195fc9693006301e91e61fbfaa62e9d8e9580508d833c8338b7e50c01c867860af1
91f44abcde81edb037188e5efe048f9fa13eb8b56cc77663763a787106a415c41ab3
c2655afd3b19903aed410809236a1a80625162617c32374a26b5a87bd2c814687421
b0595b3964a633eac4a9f43d9c51459ceb8aedbd58d096609f4f82d3dd681ad3302d
d4c875442ef615670d5aa9754f5b602c
```

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

```
1, / version /
null, / partial-iv /
null, / psa-kid /
h'18' / rx-sai from C_I /
/ message_3: h'e418...602c' /
```

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

```
a11822822e48c24ab2fd7643c79f58406deb6c6ea62b2d5101e78d64cb51b3ef2b73
d43cca717627e420550a7fef8a2a1dd61873f35805ed81e42da3f8634109b9e2ff96
3f4b92c6ba85a4278e06da5336583f010003aa0146235a91d189ea058203a1010106
f607f609010a8201010b8201010c8200000d8200000e81a2024656b44d9a538a0b4a
24ca8ebe49d4fd514649364f010101a3011904000282010203810 3
```

That plaintext has the following decoded items.

```
{34: [-15, h'C24AB2FD7643C79F']}, / ID_CRED_I with following sig. /
h'6A0E076F87CBE4AD82A6F44CDA01B4006B4C2E10AB9E7617111E9E2883B5B75BAF
F343DABFB4153D72F400726AD96E798F688393E772B28C225919B73C54D810',
-23, / EAD label with following value /
h'010003AA0146235A91D189EA058203A1010106F607F609010A8201010B8201010C
8200000D8200000E81A2024656B44D9A538A0B4A24CA8EBE49D4FD514649',
-23, / EAD label with following value /
h'010101A3011904000282010203810 3'
```

The embedded EAD_3 contains the following SAFE messages:

```
   1, / act. index /
   0, / act. step /
   3, / act. type: SC /
   {
       1: h'235A91D189EA', / sender SAI bytes /
       5: [ / KUS: /
           3, / CTX: COSE (3) /
           {
               1: 1 / alg: A128GCM (1) /
           }
       ],
       6: null, / TSI: still TBD /
       7: null, / TSR: still TBD /
       9: 1, / SMS: end-to-end (1) /
       10: [1, 1], / RX key depth range /
       11: [1, 1], / TX key depth range /
       12: [0, 0], / RX prekey depth range /
       13: [0, 0], / TX prekey depth range /
       14: [ / initial content keys: /
           {
               2: h'56B44D9A538A', / KID /
               11: h'24CA8EBE49D4FD514649' / ARN /
           }
       ]
   }

   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:

```
01f6f62d5857b9d48b055a4c2cf0d4e26bf0963dd8b76fa55d31f525eb2e8dc075e3
d993b12ac3b0f63339f6772175d1c1148c59badb8b31a4eb33f71d89b2d875b67e5c
48103d1eb7f36131bd833bd22dd81aca5be811067612e6209b
```

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

```
1, / version /
null, / partial-iv /
null, / psa-kid /
-14 / rx-sai from C_I /
/ message_3: h'b9d4...209b' /
```

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

```
364301020136583f010103aa014673134ebd4c4d058203a1010106f607f609010a82
01010b8201010c8200000d8200000e81a202469f379d2ed2f20b4af11ae27bf90480
99170c
```

That plaintext has the following decoded items.

```
-23, / EAD label with following value /
h'010201',
-23, / EAD label with following value /
h'010103aa014673134ebd4c4d058203a1010106f607f609010a8201010b8201010c
8200000d8200000e81a202469f379d2ed2f20b4af11ae27bf9048099170c'
```

The embedded EAD_4 contains the following SAFE messages:

```
1, / act. index /
2, / act. step /
1 / act. type: CI /
```

```
   1, / act. index /
   1, / act. step /
   3, / act. type: SC /
   {
       1: h'73134EBD4C4D', / local SAI bytes /
       5: [ / KUS: /
           3, / CTX: COSE (3) /
           {
               1: 1 / alg: A128GCM (1) /
           }
       ],
       6: null, / TSI: still TBD /
       7: null, / TSR: still TBD /
       9: 1, / SMS: end-to-end (1) /
       10: [1, 1], / RX key depth range /
       11: [1, 1], / TX key depth range /
       12: [0, 0], / RX prekey depth range /
       13: [0, 0], / TX prekey depth range /
       14: [ / initial content keys: /
           {
               2: h'9F379D2ED2F2', / KID /
               11: h'F11AE27BF9048099170C' / ARN /
           }
       ]
   }
```

A.5. SAFE PDU #5

The encoded PDU #5 is following byte string:

```
01410140411854e318c7cb3e935ae4f4f4df3138b0bb0a68e46b11
```

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

```
1, / version /
h'01', / partial-iv /
h'', / psa-kid /
h'18' / rx-sai from C_R /
/ message_3: h'e318...6b11' /
```

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

```
43010203
```

That plaintext has the following decoded items.

h’010203’

The embedded byte strings contain the following SAFE messages:

1, / act. index /
2, / act. step /
3 / act. type: SC /

Acknowledgments

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

Implementation Status

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

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

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

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