

Delay-Tolerant Networking
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2021

E. Birrane
S. Heiner
JHU/APL
July 10, 2020

Security Context Template
draft-birrane-dtn-scot-00

Abstract

This document defines a standard template for security contexts written for the Bundle Protocol Security Protocol (BPSec).

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 January 11, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Specification Scope	3
1.2. Related Documents	3
1.3. Terminology	4
2. System Overview	4
2.1. Methods of Establishing Context	4
2.1.1. Out-Of-Band Mode	4
2.1.2. In-Band Mode	5
2.1.3. Hybrid Mode	6
2.2. Security Policy Roles	7
2.3. Common Events and Actions	8
2.3.1. Bundle Protocol Reason Codes	8
2.3.2. Event Codes	10
2.3.3. Processing Actions	12
2.4. Security Policy Considerations	14
2.5. Security Context Template	15
2.5.1. Overview	16
2.5.2. Interfaces	16
2.5.3. Definitions	17
2.5.4. Canonicalization Algorithms	17
2.5.5. Processing	18
2.5.6. Policy	18
2.5.7. IANA Considerations	18
3. Normative References	18
Authors' Addresses	18

1. Introduction

The Bundle Protocol (BP) may operate in environments that prevent reliable synchronization of session information, to include negotiated cryptographic material. To accommodate for this possibility, BP bundles may use extension blocks to carry annotative information. The Bundle Protocol Security Protocol (BPsec) defines security-related extension blocks (security blocks) to carry cryptographic material, to optionally include material that might otherwise be expected resident at communication endpoints. To accomplish these, BPsec security blocks specify a "security context" comprising the material generated for, carried with, and/or otherwise utilized by the block.

Where BPsec security blocks traverse networks for which a synchronized security mechanism exists, a security context can be defined which minimally identifies endpoint information. For example, in networks that support TLS, a security context can be defined to carry TLS record information after a successful TLS handshake has been used to synchronize state at the endpoints of the

exchange. Alternatively, where no such synchronizing security mechanisms exist, a security context can be defined which carries necessary configuration, cryptographic material, and policy.

The diversity of networks in which BP, and thus BPsec, may be deployed implies a diversity of network contexts in which security may be applied. For this reason, it is expected that multiple security contexts will be defined for BPsec security blocks, such that BPsec agents may select the most suitable context. This document defines a template for the documentation of security contexts. This template includes Bundle Protocol (BP) reason codes, discrete events in the life-cycle of a security block, and policy actions that may be taken by a bundle node when processing a security block.

1.1. Specification Scope

This document defines the information that must be addressed in the definition of any BPsec security context specification. Specifically, this document details the following information.

- o Data specification requirements.
- o Definitions and handling requirements for standard BP reason codes.
- o Definitions and handling requirements for standard error codes.
- o Guidance for specifying context-specific parameters and results.

The SCoT addresses only that information necessary for the proper specification, interpretation, and processing of BPsec security blocks. Any specific security protocol, cipher suite, or enumeration set other than those defined by BP and BPsec are not considered part of this document.

1.2. Related Documents

This document is best read and understood within the context of the following other DTN documents:

The Concise Binary Object Representation (CBOR) format [RFC7049] defines a data format that allows for small code size, fairly small message size, and extensibility without version negotiation. The block-specific-data associated with BPsec security blocks are encoded in this data format.

The Bundle Protocol [I-D.ietf-dtn-bpbis] defines the format and processing of bundles, defines the extension block format used to represent BPsec security blocks, and defines the canonical block structure used by this specification.

The Bundle Security Protocol [I-D.ietf-dtn-bpsec] defines the BP extension blocks used to implement security services in a DTN. This also outlines the need for security contexts customized to the networks in which BP and BPsec are deployed.

1.3. Terminology

This section defines terminology unique to the Security Context Template. Definitions of other terms specific to BP and BPsec, such as "Security Acceptor", "Security Block", "Security Context", "Security Operation", "Security Service", "Security Source", and "Security Target" are as defined in BPsec [I-D.ietf-dtn-bpsec].

2. System Overview

This section describes how a security context is used by BPsec to normalize the diversity of environments encountered by the Bundle Protocol.

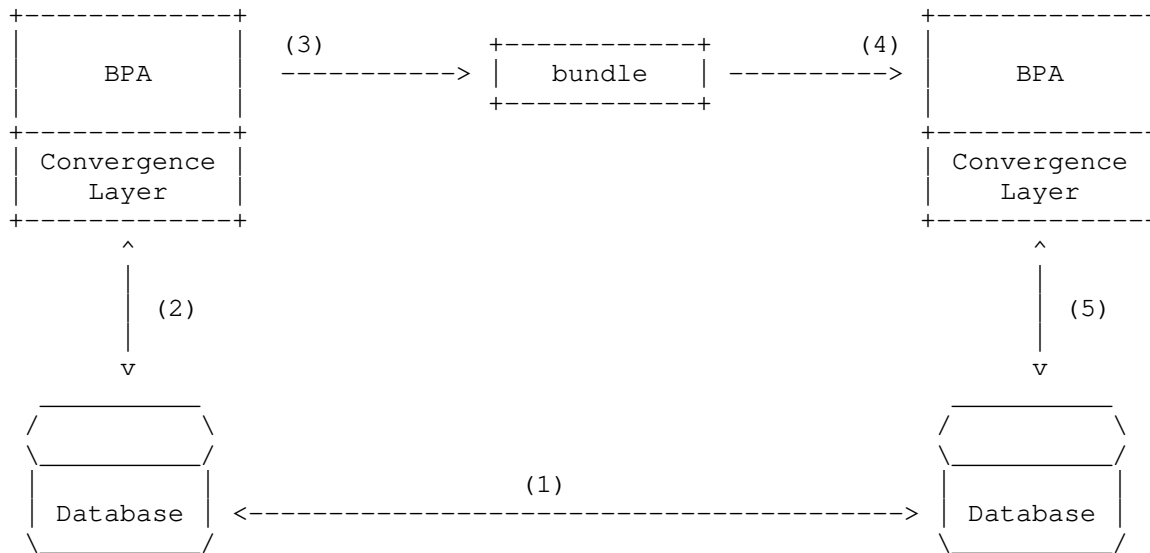
2.1. Methods of Establishing Context

The definition of a security context is based in the concept that different networks would provide annotative security information in different ways as a function of the capabilities of the network itself. This section discusses three general methods of generating context information: out-of-band, in-band, and a hybrid approach.

For each of these methods the term "in-band" refers to information carried within a bundle.

2.1.1. Out-Of-Band Mode

The Out-of-Band method of context establishment utilizes out-of-band information only, requiring session information to be pre-shared. The sending and receiving nodes must be configured using an out-of-band method such as separate key management, one-time padding, or ID-based encryption and use a pre-placed session key in order to establish the context.

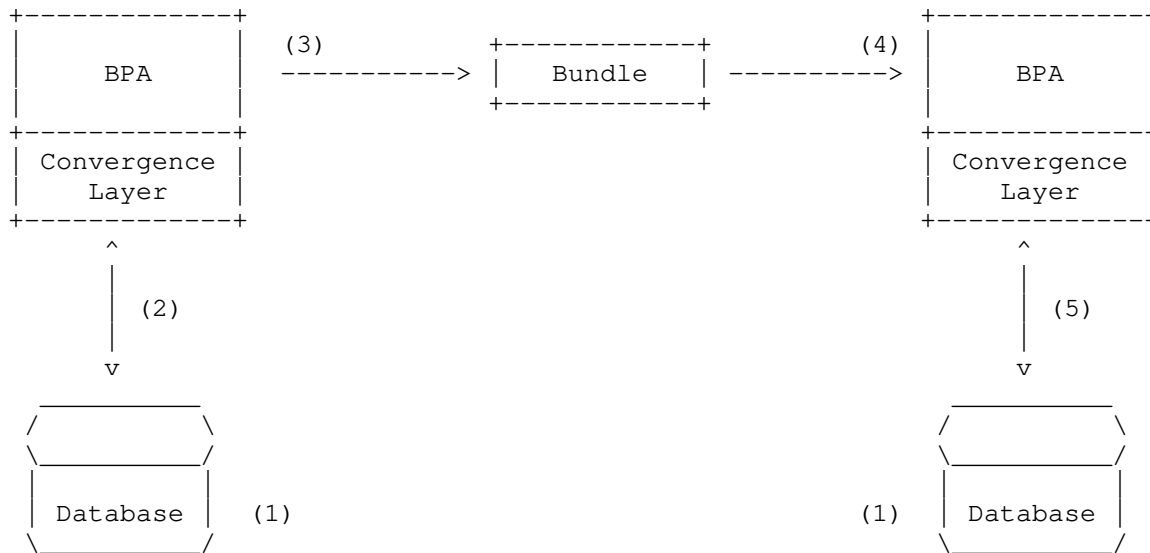


In-Band Mode

Step 1 shows the out-of-band configuration of both the sending and receiving nodes. A session key is pre-placed in the databases accessible by the nodes in step 1. Step 2 shows the sending node retrieving this session key which is used in step 3 to add a security block to the bundle which may transport a shared key between the sending and receiving nodes. The bundle arrives at the receiving node in step 4, and the pre-placed session key is retrieved in step 5. The session key can be used along with the shared key to establish the context between the two nodes.

2.1.2. In-Band Mode

The In-Band method of context establishment utilizes in-band information only. The key encryption key for both the sending and receiving node must be pre-placed so that the node can fetch the information from its in-band database. The session key and any additional information necessary to establish the context is transported by the bundle. The receiving node must use its pre-placed key-encryption-key in order to recover the session key when the bundle is received.

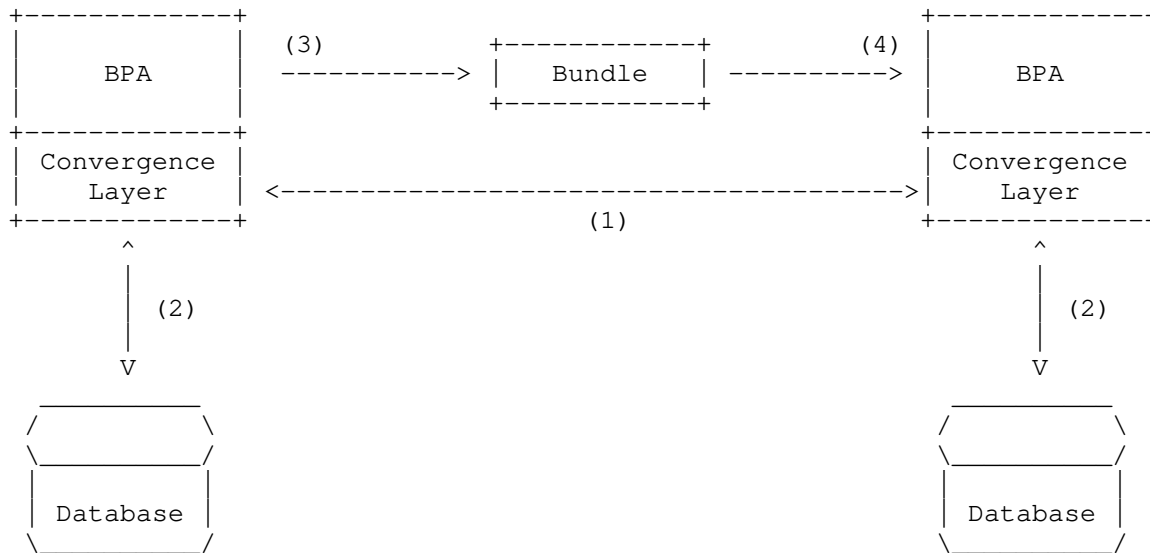


In-Band Mode

Step 1 indicates that the key encryption key has been pre-placed at both the sending and receiving node. The key encryption key is supplied to the sending node in step 2, which then uses this key in step 3 to add a security block to the bundle. This security block contains a session key and other necessary security context parameters used to establish the context. The bundle is received in step 4, and the key encryption key pre-placed at the receiving node is fetched in step 5. The key encryption key is used to recover the session key from the bundle, and the session key can be used to retrieve the rest of the security results from the block.

2.1.3. Hybrid Mode

Using the Hybrid method of context establishment, both in-band and out-of-band information is utilized. A session key is negotiated out-of-band, while any additional information necessary to establish the context is transported by the bundle.



Hybrid Mode

Step 1 shows session establishment, performed by the convergence layer of two BP nodes using in-band information including a negotiated session key. At step 2, the session data is stored at both nodes in their respective databases. The sending node adds a security block to the bundle containing the information necessary to establish the context in the form of security context parameters in step 3. Step 4 shows the augmented bundle arriving at the receiving node. The receiving node can then use the session data in its database (in-band information) and the session information transmitted in the bundle as security context parameters (out-of-band information).

2.2. Security Policy Roles

A security context may be interpreted differently based on the role of the BPsec-aware node processing the security service. This section defines the roles associated with a security operation.

Security Source

A security source node must add the security service to the bundle as specified by some policy. The bundle will first be inspected to ensure that the newly required security block has not already been added to the bundle. If it is determined that the required security block is not already represented in the bundle, a new security block is added and the specified

security context is used to apply the security service to each security target.

Security Verifier

A security verifier node must verify (not process) a security service in the bundle as specified by the receiver security policy.

A security verifier will verify the integrity signature(s) stored as security results of the BIB if the security service described by the receiver security policy rule is integrity.

If policy identifies a BCB for which the node is a security verifier, authenticity of the data belonging to the BCB's security targets is verified. Some confidentiality security contexts may not support this action, as the cipher suite(s) that they utilize do not expose an authentication function.

Security Acceptor

A security acceptor node must process a security service in the bundle as specified by policy. In order to process a security service, each security target belonging to the security block must have its integrity verified and/or its contents decrypted.

2.3. Common Events and Actions

This section describes the identification of common events and actions associated with the processing of BPsec blocks at bundle nodes. Since these items are not specific to a single security context, they should be considered standard across all defined security contexts for BPsec.

2.3.1. Bundle Protocol Reason Codes

This section describes a set of reason codes associated with the processing of a bundle based on a security analysis performed by a BPsec-aware BPA.

Bundle protocol agents (BPAs) must process blocks and bundles in accordance with both BP policy and BPsec policy. The decision to receive, forward, deliver, or delete a bundle may be communicated to the report-to address of the bundle, in the form of a status report, as a method of tracking the progress of the bundle through the network. The status report for a bundle may be augmented with a "reason code" explaining why the particular action was taken on the bundle.

Missing Security Service

This reason code indicates that a bundle was missing one or more required security services. This reason code is used when a bundle is received by a security verifier or security acceptor and is missing a security service required by that verifier or acceptor.

Unknown Security Service

This reason code indicates that a security service present in a bundle cannot be understood by the security verifier or security acceptor of that service. For example, if a security service references a security context identifier, cipher suite name, or other parameter that is not known by the verifier/acceptor then the service cannot be processed and this reason code may be sent. This reason should not be sent by a node that is not configured as a verifier or acceptor of the service. There is no requirement that all BPSec-aware nodes in a network be able to understand all security services in all bundles in the network.

Unexpected Security Service

This reason code indicates that a BPSec-aware node received a bundle which contained more security services than expected. This is typically used to indicate that a bundle contained a security service which was not required by the BPSec-aware node receiving the bundle. This reason code should not be seen as an error condition as it is expected that BPSec-aware nodes will see security services in a bundle for which they are neither a verifier nor an acceptor. In certain networks, this reason code may be useful in identifying misconfiguration in the network.

Failed Security Service

This reason code indicates that a BPSec-aware node was unable to process an existing, known security service in a bundle. This may occur when a security-source is unable to add a required service to a bundle. This may occur if a security service fails to verify at a security verifier. This may occur if a security service fails to be processed at a security acceptor.

Conflicting Security Services

This reason code indicates that a bundle received by a BPSec-aware node included a set of security services disallowed by the BPSec protocol and that security processing was unable to proceed because of a BPSec protocol violation.

2.3.2. Event Codes

A life-cycle of a security operation within BPSec is independent of the security context used to populate the contents of that security operation. However, security contexts must provide guidance on how a BPSec-aware node should react to these events for security operations using that context.

This section identifies the unique events in the life-cycle of a security operation that may identify processing points within a security context.

2.3.2.1. Security Source Events

At a security source, three events may be associated with the security operation as its life-cycle is initiated.

`source_for_sop`

When a node is designated as a security source for a security operation, there is a security policy rule that requires the security operation to be present in the bundle. When the sender security policy rule that applies to the bundle is identified, the security operation is acknowledged as needed.

`sop_added_at_source`

A security operation is added when it is represented in the bundle as a fully populated security block. In order for the security operation to be considered as added, the security block must be allocated for the bundle, represented in the bundle, and populated. Population of a security block includes information provided by the sender security policy rule and any security results associated with the security operation. These security results must be calculated and stored in either the security block or the security target block's block-type-specific data.

`sop_misconfigured_at_source`

When a security operation is transitioning from being needed to added, it may end up misconfigured. A security operation may be misconfigured if resource exhaustion occurs and there is not appropriate space to store a required security block or other components of the security block. A security operation will also be considered to be misconfigured if any of the required security results cannot be successfully calculated.

2.3.2.2. Security Verifier Events

At a security verifier, five events may be associated with the security operation.

`verifier_for_sop`

A node is designated as a security verifier when a receiver security policy rule is identified which is applicable to the current bundle and is associated with the security verifier role. The security operation described by the rule is then considered to be needed by the security verifier node.

`sop_misconfigured_at_verifier`

A security operation may be identified as misconfigured by the security verifier node. A misconfigured security operation may encounter a resource allocation issue and be unable to add an additional, required security result. A misconfigured security operation may also identify an incorrect security context or parameter, causing a conflict the security policy rule.

`sop_missing_at_verifier`

A security operation may transition from needed to missing if the required security block cannot be located in the bundle by the security verifier node.

`sop_corrupted_at_verifier`

A corrupted security operation is identified as a security block which is unsuccessfully processed by the security verifier node. A corrupted security operation indicates that the security target cannot be verified.

`sop_verified`

When a security operation is designated as processed, the security target has been successfully verified.

2.3.2.3. Security Acceptor Events

At a security acceptor, five events may be associated with the security operation.

`acceptor_for_sop`

A node is designated as a security acceptor when a receiver security policy rule is identified that is both applicable to the current bundle and associated with the security acceptor role. The security operation described by the rule is then considered to be needed by the security acceptor node.

`sop_misconfigured_at_acceptor`

A security operation may be identified as misconfigured by the security acceptor node. A misconfigured security operation signals an incorrect security context or parameter, causing a conflict the security policy rule.

sop_missing_at_acceptor

A security operation is missing if the required security block cannot be located in the bundle by the security acceptor node.

sop_corrupted_at_acceptor

A corrupted security operation is identified as a security block which is unsuccessfully processed by the security acceptor node. A corrupted security operation indicates that the security target cannot be verified and/or decrypted.

sop_processed

When a security operation is designated as processed, the security target has been successfully verified and/or decrypted and is removed from the bundle.

2.3.3. Processing Actions

This section defines a standard set of processing actions that can be specified when defining the policy associated with a security context. The benefit of enumerating these actions is to provide a common set of terminology and design across multiple security contexts with the purpose of making the development of multi-security-context BPSec implementations as similar as possible.

In particular, a security context may wish to override how a BPSec implementation treats the block processing control flags associated with a security block and/or its target block using that context. While the creator of a target block may set block processing flags it may be insecure to process the block in accordance with those flags. Similarly, if a target block has been modified since its was created, it is possible that the target block's processing flags have also been modified and should not necessarily be honored by a receiving BPA.

A security context should specify the situations in which the following actions should be taken by a BPSec implementation at a BPSec-aware node in the network.

remove_sop

When the security operation is removed, it is no longer required for that bundle. If the security operation is the only operation represented by the security block, the block must be removed from the bundle. Otherwise, the security

target's block number is removed from the security block to indicate that the particular operation no longer applies.

`remove_sop_target`

Remove security operation's security target and the security operations associated with that target - The security operation's security target is removed from the bundle, as are any additional security operations associated with that target. For example, if the target block of a confidentiality service is removed, the integrity security operation for that target would be removed as well.

`remove_all_target_sops`

Remove all security operations for the security target - This option is used to remove all of the security operations associated with the security target while retaining the security target in the bundle.

`do_not_forward`

Selecting this option will end bundle transmission at the current node, regardless of bundle destination.

`request_storage`

This option is paired with 'Do Not Forward Bundle' in order to retain a copy of the bundle at the current node. Bundle storage cannot be guaranteed, as node resources are unknown at the time of bundle transmission, but selecting this option will result in an effort to retain a copy of the bundle at the node.

`report_reason_code(code)`

Generate a status report describing the treatment of the bundle and use the provided reason code as the reason.

`override_target_bpcf(mask, new_values)`

Override one or more block processing control flags of the target block and process the block in accordance with the overridden flags. The overridden flags MUST NOT be written back to the target block, and only used for processing the block locally.

Manipulating individual flags to be forced to 0, forced to 1, or kept as specified in the target block requires two inputs. This can be accomplished by the operation:

```
local_flags = (block_flags & mask) | (~mask & values)
```

`override_sop_bpcf(mask, new_values)`

Override one or more block processing control flags of the security block associated with the sop and process the block in accordance with the overridden flags. The overridden flags MUST NOT be written back to the security block, and only used for processing the block locally.

Manipulating individual flags to be forced to 0, forced to 1, or kept as specified in the target block requires two inputs. This can be accomplished by the operation:

```
local_flags = (block_flags & mask) | (~mask & values)
```

2.4. Security Policy Considerations

A security context details the environment in which cryptographic materials are provided to, and retrieved from, the cipher suites used for security services in the network. For this reason, the policy associated with determining proper configuration information must be addressed in the specifications defining new security contexts because this information may differ amongst security contexts.

This section identifies several policy questions that should be addressed in the specification of a security context. In the absence of guidelines for a specific security context, this section defines what should be considered default behavior for any security context.

Target Block Identification

Security contexts should define how a target block of a security service should be identified. This identification is used when determining whether to add a security block at a security source, whether to check a security block at a verifier, and whether a node should consider itself the acceptor of the block.

DEFAULT BEHAVIOR: Unless otherwise specified, a target block MUST be identified at a security source by the three-tuple of {bundle source EID, bundle destination EID, and block type of the target block}. A target block at a security verifier and a security acceptor is identified by these three information elements plus the security context identifier.

Security Parameter Local Overrides

Security contexts should define the circumstances in which a local BPsec-aware node may override parameters in a security block with locally-configured parameters.

DEFAULT BEHAVIOR: Unless otherwise specified, a locally-provided security parameter MUST NOT override a security

parameter present in a security block. A local parameter can only be used in cases where the corresponding parameter is not present in the security block itself.

Security Processing Actions

Security contexts should define the circumstances in which the processing actions defined in Section 2.3.3 should be used and with what inputs.

DEFAULT BEHAVIOR: Unless otherwise specified, the local, BPSec-aware node MUST enforce policy actions as a function of some local policy definition at the node itself. This may be through the definition of generic actions for all security contexts or it may be identified based on a specific security context.

2.5. Security Context Template

This section defines a recommended outline for the definition of a security context for BPSec. The purpose of such an outline is to both ensure that no critical information is omitted when authoring new security contexts and to reduce the cognitive load associated with implementing new security contexts by having them conform to a standard representation.

A security context specification should include the following elements.

1. Overview
2. Interfaces
3. Definitions
4. Canonicalization Algorithms
5. Processing
6. Policy
7. IANA Considerations

A single specification may define one or more security contexts, in which case this information would be repeated for each security context.

The remainder of this section provides information on the topics that should be covered in each of these sections.

2.5.1. Overview

This section should identify the rationale for creating a security context, whether the context uses in-band, out-of-band, or hybrid mechanisms for information exchange, and the unique situations in which the context should be used.

2.5.2. Interfaces

Security context interfaces detail any special considerations or assumptions made when designing the algorithms for creating or otherwise processing the contents of security blocks.

2.5.2.1. Information Provided to the Cipher Suite

This section should identify how the security context interfaces with the cipher suite (or cipher suites) used to process the security service. This may be as simple as providing parameters from the security block to a cipher suite implementation. This may also be a more complex interface involving fusing parameter information carried by the security block with local information to provide an interface to the cipher suite.

A cipher suite may include parameters such as a key length, mode, or number of rounds, which is part of the cipher suite definition. These parameters are considered "locked" parameters and are not to be confused with the "free" parameters that the user may define for each security context. Where not already defined by the cipher suite itself, the security context should clearly identify which parameters should be considered "free" and which should be considered "locked".

In addition to listing cipher suites, this section should identify information relating to all "free" cipher suite parameters. This may include selected bit lengths, which parameters are provided by the local node, which parameters must be present in the security block itself, and how parameters should be protected when represented in a security block.

2.5.2.2. Information Provided by the BPA

A security context specification should describe the information it expects to be provided to it by the local BPA, separate from the contents of the security block and target block from the bundle.

For example, a BPA may additionally provide local the security context parameters.

2.5.2.3. Information Provided to the BPA

A security context specification should describe the information that it will provide back to the local BPA. Typically this is either the output of a cipher suite to be added to a security block in a bundle, or some signal for a process event associated with a verification or processing action.

2.5.3. Definitions

This section defines custom parameters and results associated with the security context and carried either within a security block or as part of local node configuration.

Each security context parameters and result is defined as a key-value pair, where the key is a CBOR-encoded integer and the value is defined as the CBOR encoding of its data type. This section must define any unique parameters and results used by the security context, to include the following information.

- o The integer identifier of the parameter or result item.
- o Whether multiple instances of this item can be present in a security block at one time, and whether at least one instance of this item is required to be defined.
- o Whether this item must be present only in the security block, or whether it can also be included as part of the configuration of a local node, and how to determine which version of the item to use in the event that it is defined in both the security block and the local node.
- o The data type of the item and how that data type must be CBOR encoded for representation in the security block.

2.5.4. Canonicalization Algorithms

Security context canonicalization algorithms take precedence over any others defined. The exact data and its form when provided to the canonicalization algorithm must be determined by the security context.

Consistency and determinism of the block-type-specific data provided as input to the security context is critical for security services to function as expected.

2.5.5. Processing

A security context specification should describe how the context should be used to perform every processing action described in Section 2.3.2.

2.5.6. Policy

A security context specification should describe how the context should be configured from a policy perspective. This should include, at a minimum, under which circumstances each policy action identified in Section 2.3.3 should be taken.

2.5.7. IANA Considerations

Each security context MUST provide an entry in the IANA security context identifier registry to uniquely identify the identifiers of each security context.

3. Normative References

[I-D.ietf-dtn-bpbis]

Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol Version 7", draft-ietf-dtn-bpbis-25 (work in progress), May 2020.

[I-D.ietf-dtn-bpsec]

Birrane, E. and K. McKeever, "Bundle Protocol Security Specification", draft-ietf-dtn-bpsec-22 (work in progress), March 2020.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.

Authors' Addresses

Edward J. Birrane, III
The Johns Hopkins University Applied
Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
US

Phone: +1 443 778 7423
Email: Edward.Birrane@jhuapl.edu

Sarah Heiner
The Johns Hopkins University Applied
Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
US

Phone: +1 240 592 3704
Email: Sarah.Heiner@jhuapl.edu