DTN Research Group   S. Symington
Internet-Draft   The MITRE Corporation
Expires: March 23, 2008   S. Farrell
                                            Trinity College Dublin
                                                            H. Weiss
                                                           P. Lovell
                                                        SPARTA, Inc.
                                                 September 20, 2007

Bundle Security Protocol Specification
draft-irtf-dtnrg-bundle-security-04

Status of this Memo

Copyright Notice

Abstract

   This document defines the bundle security protocol, which provides
   data integrity and confidentiality services. We also describe
   various bundle security considerations including policy options.

Table of Contents

1. Introduction

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [1].

This document defines security features for the bundle protocol [2] intended for use in delay tolerant networks, in order to provide the DTN security services as described in the DTN Security Overview and Motivations document [8].

The bundle protocol is used in DTNs which overlay multiple networks, some of which may be challenged by limitations such as intermittent and possibly unpredictable loss of connectivity, long or variable delay, asymmetric data rates, and high error rates. The purpose of the bundle protocol is to support interoperability across such stressed networks. The bundle protocol is layered on top of underlay-network-specific convergence layers, on top of network-specific lower layers, to enable an application in one network to communicate with an application in another network, both of which are spanned by the DTN.

Security will be important for the bundle protocol. The stressed environment of the underlying networks over which the bundle protocol will operate makes it important that the DTN be protected from unauthorized use, and this stressed environment poses unique challenges on the mechanisms needed to secure the bundle protocol. Furthermore, DTNs may very likely be deployed in environments where a portion of the network might become compromised, posing the usual security challenges related to confidentiality, integrity and availability.

1.1. Related Documents

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

The Delay-Tolerant Network Architecture [9] defines the architecture for delay-tolerant networks, but does not discuss security at any length.

The DTN Bundle Protocol [2] defines the format and processing of the blocks used to implement the bundle protocol, excluding the security-specific blocks defined here.

The Delay-Tolerant Networking Security Overview [8] provides an informative overview and high-level description of DTN security.

. **Terminology**

   We introduce the following terminology for purposes of clarity:
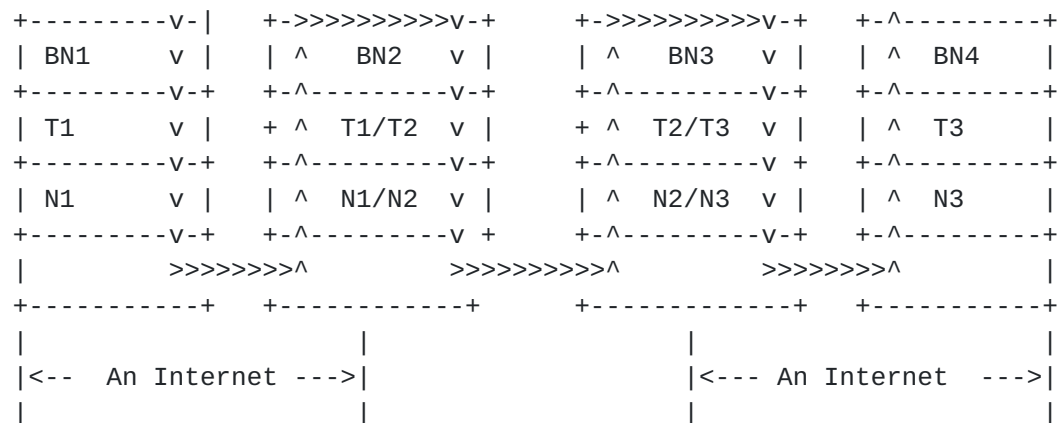
      source - the bundle node from which a bundle originates

      destination - the bundle node to which a bundle is ultimately
      destined

      forwarder - the bundle node that forwarded the bundle on its most
      recent hop

      intermediate receiver or "next hop" - the neighboring bundle node
      to which a forwarder forwards a bundle.

   In the figure below, which is adapted from figure 1 in the Bundle
   Protocol Specification, four bundle nodes (denoted BN1, BN2, BN3, and
   BN4) reside above some transport layer(s). Three distinct transport
   and network protocols (denoted T1/N1, T2/N2, and T3/N3) are also
   shown.

```
+---------v-|    +->>>>>>>>>>v-+     +->>>>>>>>>>v-+    +-^---------+
| BN1     v |    | ^   BN2   v |     | ^   BN3   v |    | ^  BN4    |
+---------v-+    +-^---------v-+     +-^---------v-+    +-^---------+
| T1      v |    + ^  T1/T2  v |     + ^  T2/T3  v |    | ^   T3    |
+---------v-+    +-^---------v-+     +-^---------v +    +-^---------+
| N1      v |    | ^  N1/N2  v |     | ^  N2/N3  v |    | ^   N3    |
+---------v-+    +-^---------v +     +-^---------v-+    +-^---------+
|       >>>>>>>>>^        >>>>>>>>>>^        >>>>>>>>^           |
+-----------+   +-----------+      +------------+   +-----------+
|               |                  |                |             |
|<--  An Internet --->|                 |<--- An Internet  --->|
|               |                  |                |             |
```

   BN = "Bundle Node" (as defined in the Bundle Protocol Specification

   Bundle Nodes Sit at the Application layer of the Internet Model.

                              Figure 1

   Bundle node BN1 originates a bundle that it forwards to BN2. BN2
   forwards the bundle to BN3, and BN3 forwards the bundle to BN4. BN1
   is the source of the bundle and BN4 is the destination of the bundle.
   BN1 is the first forwarder, and BN2 is the first intermediate
   receiver; BN2 then becomes the forwarder, and BN3 the intermediate

receiver; BN3 then becomes the last forwarder, and BN4 the last
intermediate receiver, as well as the destination.

If node BN2 originates a bundle (for example, a bundle status report
or a custodial signal), which is then forwarded on to BN3, and then
to BN4, then BN2 is the source of the bundle (as well as being the
first forwarder of the bundle) and BN4 is the destination of the
bundle (as well as being the final intermediate receiver).

We introduce the following security-specific DTN terminology:

   security-source - a bundle node that adds a security block to a
   bundle

   security-destination - a bundle node that processes a security
   block of a bundle

   security zone - that part of the network path from the security-
   source to the security-destination

Referring to Figure 1 again:

If the bundle that originates at BN1 as source is given a security
block by BN1, then BN1 is the security-source of this bundle with
respect to that security block, as well as being the source of the
bundle.

If the bundle that originates at BN1 as source is given a security
block by BN2, then BN2 is the security-source of this bundle with
respect to that security block, even though BN1 is the source.

If the bundle that originates at BN1 as source is given a security
block by BN1 that is intended to be processed by BN3, then BN1 is the
security-source and BN3 is the security destination with respect to
this security block. The security zone for this block is BN1 to BN3.

A bundle may have multiple security blocks. The security-source of a
bundle with respect to a given security block in the bundle may be
the same as or different from the security-source of the bundle with
respect to a different security block in the bundle. Similarly, the
security-destination of a bundle with respect to each of that
bundle's security blocks may be the same or different. Therefore the
security zones for various blocks may be and often will be different.

If the bundle that originates at BN1 as source is given a security
block by BN1 that is intended to be processed by BN3, and BN2 adds a
security block with security-destination BN4, the security zones for
the two blocks overlap but not completely. This problem is discussed

further in [Section 3.3](#).

Forwarding nodes MUST transmit blocks in the same order as they were
received. This requirement applies to all dtn nodes, not just ones
which implement security processing. Blocks in a bundle may be added
or deleted according to the applicable specification, but those
blocks which are both received and transmitted MUST be transmitted in
the same order that they were received.

The block sequence also indicates the order in which certain
significant actions have affected the bundle, and therefore the
sequence in which actions must occur in order to produce the bundle
at its destination.

2. **Security Blocks**

   There are three types of security block that MAY be included in a
   bundle. These are the Bundle Authentication Block (BAB), the Payload
   Security Block (PSB), and the Confidentiality Block (CB).

      The BAB is used to assure the authenticity and integrity of the
      bundle along a single hop from forwarder to intermediate receiver.

      The PSB is used to assure the authenticity and integrity of the
      bundle from the PSB security-source, which creates the PSB, to the
      PSB security-destination, which verifies the PSB authenticator.
      The authentication information in the PSB may (if the ciphersuite
      allows) be verified by any node inbetween the PSB security-source
      and the PSB security-destination that has access to the
      cryptographic keys and revocation status information required to
      do so.

      Since a BAB protects a bundle on a "hop-by-hop" basis and a PSB
      protects on a (sort of) "end-to-end" basis, whenever both are
      present the BAB MUST form the "outer" layer of protection - that
      is, the BAB MUST always be calculated and added to the bundle
      after the PSB has been calculated and added to the bundle.

      The CB indicates that some parts of the bundle have been encrypted
      at the CB security-source in order to protect the bundle content
      while in transit to the CB security-destination.

   Each of the security blocks uses the Canonical Bundle Block Format as
   defined in the Bundle Protocol Specification. That is, each security
   block is comprised of the following elements:

      - Block type code

      - Block processing control flags

      - Block EID reference list (optional)

      - Block data length

      - Block-type-specific data fields

   Since the three security blocks have most fields in common, we can
   shorten the description of the Block-type-specific data fields of
   each security block if we first define an abstract security block
   (ASB) and then specify each of the real blocks in terms of the fields
   which are present/absent in an ASB. Note that no bundle ever
   contains an ASB, which is simply a specification artifact.

2.1. Abstract Security Block

   An ASB consists of the following mandatory and optional fields:

      - Block-type code (one byte) - as in all bundle protocol blocks
      except the primary bundle block. The block types codes for the
      security blocks are:

         BAB: 0x02

         PSB: 0x03

         CB: 0x04

      - Block processing control flags (SDNV) - defined as in all bundle
      protocol blocks except the primary bundle block (as described in
      the Bundle Protocol [2]). SDNV encoding is described in the
      bundle protocol. There are no general constraints on the use of
      the block processing flags, and some specific requirements are
      discussed later.[Comment.1][Comment.2]

      - EID references - composite field defined in [2] containing
      references to one or two EIDs. Presence of the EID-reference
      field is indicated by the setting of the "block contains an EID-
      reference field" (EID_REF) bit of the block processing control
      flags. If one or more references is present, flags in the
      ciphersuite ID field, described below, specify which.

      If no EID fields are present then the composite field itself is
      omitted entirely, rather than containing a count field of zero,
      since such a representation is not permitted. The EID_REF bit is
      not set.

      The possible EIDs are:-

         - (optional) Security-source - specifies the security source
         for the service. If this is omitted, then the source of the
         bundle is assumed to be the security-source.

         - (optional) Security-destination - specifies the security
         destination for the service. If this is omitted, then the
         destination of the bundle is assumed to be the security-
         destination.

      If two EIDs are present, security-source is first and security-
      destination comes second.

- Block data length (SDNV) - as in all bundle protocol blocks
except the primary bundle block. SDNV encoding is described in
the bundle protocol.

- Block-type-specific data fields as follows:

    - Ciphersuite ID (SDNV)

    - Ciphersuite flags (SDNV)

    - (optional) Correlator - when more than one related block is
    inserted then this field MUST have the same value in each
    related block instance. This is encoded as an SDNV. See note
    in Section 3.8 with regard to correlator values in bundle
    fragments.

    - (optional) Ciphersuite parameters - compound field of next
    two items

        - Ciphersuite parameters length - specifies the length of
        the following Ciphersuite parameters data field and is
        encoded as an SDNV.

        - Ciphersuite parameters data - parameters to be used with
        the ciphersuite in use, e.g. a key identifier or
        initialization vector (IV). See Section 2.5 for a list of
        potential parameters and their encoding rules. The
        particular set of parameters that are included in this field
        are defined as part of the ciphersuite specification.

    - (optional) Security result - compound field of next two items

        - Security result length - contains the length of the next
        field and is encoded as an SDNV.

        - Security result data - contains the results of the
        appropriate ciphersuite-specific calculation (e.g. a
        signature, MAC or ciphertext block key).

```
+----------------+---------------+---------------+---------------+
| type           | flags (SDNV)  | EID ref list(comp)            |
+----------------+---------------+---------------+---------------+
| length  (SDNV)                 | ciphersuite (SDNV)            |
+----------------+---------------+---------------+---------------+
| ciphersuite flags (SDNV)       | correlator  (SDNV)            |
+----------------+---------------+---------------+---------------+
|params len(SDNV)| ciphersuite params data                      |
+----------------+---------------+---------------+---------------+
|res-len  (SDNV) | security result data                         |
+----------------+---------------+---------------+---------------+
```

The structure of an abstract security block

Figure 2

Some ciphersuites are specified in Section 4, which also specifies
the rules which MUST be satisfied by ciphersuite specifications.
Additional ciphersuites MAY be defined in separate specifications.
Ciphersuite IDs not specified are reserved. Implementations of the
bundle security protocol decide which ciphersuites to support,
subject to the requirements of section Section 4. It is RECOMMENDED
that implementations that allow additional ciphersuites permit
ciphersuite ID values at least up to and including 127, and they MAY
decline to allow larger ID values.

The structure of the ciphersuite flags field is shown in Figure 3.
In each case the presence of an optional field is indicated by
setting the value of the corresponding flag to one. A value of zero
indicates the corresponding optional field is missing. Presently
there are five flags defined for the field and for convenience these
are shown as they would be extracted from a single-byte SDNV. Future
additions may cause the field to grow to the left so, as with the
flags fields defined in [2], the description below numbers the bit
positions from the right rather than the standard RFC definition
which numbers bits from the left.

   src - bit 4 indicates whether the EID-reference field of the ASB
   contains the optional reference to the security-source.

   dest - bit 3 indicates whether the EID-reference field of the ASB
   contains the optional reference to the security-destination.

   parm - bit 2 indicates whether the ciphersuite-parameters-length
   and ciphersuite parameters data fields are present or not.

   corr - bit 1 indicates whether or not the ASB contains an optional
   correlator.

res - bit 0 indicates whether or not the ASB contains the security
result length and security result data fields.

bits 5-6 are reserved for future use.

Ciphersuite flags
   Bit   Bit   Bit   Bit   Bit   Bit   Bit
    6   5   4   3   2   1   0
  +-----+-----+-----+-----+-----+-----+-----+
  | reserved |src |dest |parm |corr |res |
  +-----+-----+-----+-----+-----+-----+-----+

Figure 3

A little bit more terminology: if the block is a PSB then when we
refer to the "PSB-source", we mean the security source for the PSB as
represented by the EID reference in the EID-references field.
Similarly we may refer to the CB-dest, meaning the security-
destination of the CB, again as represented by an EID reference. For
example, referring to Figure 1 again, if the bundle that originates
at BN1 as source is given a Confidentiality Block (CB) by BN1 that is
protected using a key held by BN3 and it is given a Payload Security
Block (PSB) by BN1, then BN1 is both the CB-source and the PSB-source
of the bundle, and BN3 is the CB-dest of the bundle.

The correlator field is used to associate several related instances
of a security block. This can be used to place a BAB that contains
the ciphersuite information at the "front" of a (probably large)
bundle, and another correlated BAB that contains the security result
at the "end" of the bundle. This allows even very memory-constrained
nodes to be able to process the bundle and verify the BAB. There are
similar use cases for multiple related instances of PSB and CB as
will be seen below.

The ciphersuite specification MUST make it clear whether or not
multiple block instances are allowed, and if so, under what
conditions. Some ciphersuites can of course leave flexibility to the
implementation, whereas others might mandate a fixed number of
instances.

2.2. Bundle Authentication Block

In this section we describe typical BAB field values for two
scenarios - where a single instance of the BAB contains all the
information and where two related instances are used, one "up front"
which contains the ciphersuite and another following the payload

which contains the security result (e.g. a MAC).

For the case where a single BAB is used:

   The block-type code field value MUST be 0x02.

   The block processing control flags value can be set to whatever
   values are required by local policy.

   The ciphersuite ID MUST be documented as a hop-by-hop
   authentication-ciphersuite which requires one instance of the BAB.

   The correlator field MUST NOT be present.

   The ciphersuite parameters field MAY be present, if so specified
   in the ciphersuite specification.

   An EID reference to the security-source SHOULD be present and, if
   so, it MUST identify the forwarder of the bundle. (If the
   forwarding node is identified in another block of the bundle that
   the next hop supports, e.g., the Previous Hop Insertion Block, the
   forwarding node need not be identified in the BAB. Similarly, if
   the forwarding node is known through other implementation-specific
   means, such as from the convergence layer, an EID reference need
   not be included.)

   An EID reference to the security-destination SHOULD NOT be present
   unless the ciphersuite requires this information (since the first
   node receiving the bundle ought be the one to validate the BAB).

   The security result MUST be present as it is effectively the
   "output" from the ciphersuite calculation (e.g. the MAC or
   signature) applied to the (relevant parts of) the bundle (as
   specified in the ciphersuite definition).

For the case using two related BAB instances, the first instance is
as defined above, except the ciphersuite ID MUST be documented as a
hop-by-hop authentication ciphersuite that requires two instances of
the BAB. In addition, the correlator MUST be present and the
security result length and security result fields MUST be absent.
The second instance of the BAB MUST have the same correlator value
present and MUST contain security result length and security result
data fields. The other optional fields MUST NOT be present.
Typically, this second instance of a BAB will be the last block of
the bundle.

2.3. Payload Security Block

A PSB is an ASB with the following additional restrictions:

The block type code value MUST be 0x03.

The block processing control flags value can be set to whatever values are required by local policy.

The ciphersuite ID MUST be documented as an end-to-end authentication-ciphersuite or as an end-to-end error-detection-ciphersuite.

The correlator MUST be present if the ciphersuite requires more than one related instance of a PSB be present in the bundle. The correlator MUST NOT be present if the ciphersuite only requires one instance of the PSB in the bundle.

The ciphersuite parameters field MAY be present.

An EID reference to the security-source MAY be present.

An EID reference to the security-destination MAY be present.

The security result is effectively the "output" from the ciphersuite calculation (e.g. the MAC or signature) applied to the (relevant parts of) the bundle. As in the case of the BAB, this field MUST be present if the correlator is absent. If more than one related instance of the PSB is required then this is handled in the same way as described for the BAB above.

The ciphersuite may process less than the entire original bundle payload, either because the current payload is a fragment of the original bundle or just becuase it is defined to process some subset. For whatever reason, if the ciphersuite processes less than the complete, original bundle payload MUST specify, as part of the ciphersuite parameters, which bytes of the bundle payload are protected.

For some ciphersuites, (e.g. those using asymmetric keying to produce signatures or those using symmetric keying with a group key), the security information can be checked at any hop on the way to the destination that has access to the required keying information. This possibility is further discussed in Section 3.6 below.

Most asymmetric PSB-ciphersuites will use the PSB-source to indicate the signer and will not require the PSB-dest field because the key needed to verify the PSB authenticator will be a public key

associated with the PSB-source.

2.4. Confidentiality Block

A typical confidentiality ciphersuite will encrypt the payload using
a randomly generated bundle encrypting key (BEK) and will use a CB
security result to carry the BEK encrypted with some long term key
encryption key (KEK) or well-known public key. If neither the
destination nor security-destination resolves the key to use for
decryption, the ciphersuite parameters field can be used to indicate
the decryption key with which the BEK can be recovered. Subsequent
CB security results will contain blocks encrypted using the BEK if
non-payload blocks are to be encrypted.

It is STRONGLY RECOMMENDED that a data integrity mechanism be used in
conjunction with confidentiality, and that encryption-only
ciphersuites NOT be used. The "authentication tag" or "integrity
check value" is stored into security-result rather than being
appended to the payload as is common in some protocols since, as
described below, it is important that there be no change in the size
of the payload.

The payload is encrypted "in-place", that is, following encryption,
the payload block payload field contains ciphertext, not plaintext.
The payload block processing flags are unmodified.[Comment.3]

The "in-place" encryption of payload bytes is to allow bundle payload
fragmentation and re-assembly, and custody transfer, to operate
without knowledge of whether or not encryption has occurred and, if
so, how many times.

Fragmentation and reassembly and custody transfer are adversely
affected by a change in size of the payload due to ambiguity about
what byte range of the original payload is actually in any particular
fragment. Ciphersuites SHOULD place any payload expansion, such as
authentication tags (integrity check values) and any padding
generated by a block-mode cipher, into an "integrity check value"
item in the security-result field (see Section 2.5) of the
confidentiality block.

Payload super-encryption is allowed. The second application of
confidentiality does not generally protect the parameters of the
first which represent a vulnerability in some circumstances.
Ciphersuites SHOULD define super-encryption such that it protects the
parameters of earlier encryption.

Confidentiality can be applied to blocks other than the payload and
in some cases this is recommended. The ciphersuite may apply

different rules to confidentiality for non-payload blocks.

For example, multiple related CB instances are required if both the
payload and the PS blocks in the bundle are to be encrypted. These
multiple CB instances require correlators to associate them with each
other.

There are situations where more than one CB instance is required but
the instances are not "related" in the sense which requires
correlators. One example is where a payload is encrypted for more
than one security-destination so as to be robust in the face of
routing uncertainties. These multiple CB instances, are not
"related" and should not contain correlators. [Comment.4]

A CB is an ASB with the following additional restrictions:

   The block type code value MUST be 0x04.

   The block processing control flags value can be set to whatever
   values are required by local policy, except that a CB containing
   an encrypted key or a key reference MUST have the "replicate in
   every fragment" flag set. This flag SHOULD NOT be set otherwise.

   The ciphersuite ID MUST be documented as a confidentiality-
   ciphersuite.

   The correlator MUST be present if there is more than one related
   CB instance. The correlator MUST NOT be present if there are no
   related CB instances.

   If a correlator is present, the encrypted key or key identifier
   MUST be placed in the first CB of the correlated set.

   Any additional bytes generated as a result of encryption and/or
   authentication processing of the payload SHOULD be placed in an
   "integrity check value" field (see Section 2.5) in the security-
   result of the first CB.

   The ciphersuite parameters field MAY be present.

   An EID reference to the security-source MAY be present.

   An EID reference to the security-destination MAY be present.

   The security result MAY be present and normally contains fields
   such as an encrypted bundle encryption key, authentication tag or
   the encrypted versions of bundle blocks other than the payload
   block.

As was the case for the BAB and PSB, if the ciphersuite requires more
than one instance of the CB, then the first occurrence MUST contain
any optional fields (e.g. security destination etc.) that apply to
all instances with this correlator. These MUST be contained in the
first instance and MUST NOT be repeated in other correlated blocks.
Fields that are specific to a particular instance of the CB MAY
appear in that CB. For example, security result fields MAY (and
probably will) be included in multiple related CB instances, with
each result being specific to that particular block. Similarly,
several CBs might each contain a ciphersuite parameters field with an
IV specific to that CB instance.

Put another way: when confidentiality will generate multiple blocks,
it MUST first create a CB with the required ciphersuite ID,
parameters etc. as specified above. Typically, this CB will appear
"early" in the bundle. If this "first" CB doesn't contain all of the
ciphertext, then it may be followed by other, correlated CBs which
MUST NOT repeat the ciphersuite parameters, security-source, or
security-destination fields from the first CB.

A CB ciphersuite may, or may not, specify which blocks are to be
encrypted. If the ciphersuite doesn't specify this, then the node is
free to encrypt whichever blocks it wishes. If a CB ciphersuite does
specify which blocks are to be encrypted, then doing otherwise is an
error.

The ciphersuite may process less than the entire original bundle
payload, either because the current payload is a fragment of the
original bundle or just becuase it is defined to process some subset.
For whatever reason, if the ciphersuite processes less than the
complete, original bundle payload MUST specify, as part of the
ciphersuite parameters, which bytes of the bundle payload are
protected.

After decryption the recovered plaintext should then replace the CB
in the bundle for further processing (e.g. PSB verification). This
recovered plaintext MUST contain all the appropriate block type,
processing flags and length information. In other words delete the
CB in question and place the recovered plaintext, which consists of
some complete non-payload block, in the bundle at the location from
which the CB was deleted.

A ciphersuite MUST NOT mix payload data and a non-payload block in a
single CB.

Even if a to-be-encrypted block has the "discard" flag set, whether
or not the CB's "discard" flag is set is an implementation/policy
decision for the encrypting node. (The "discard" flag is more

properly called the "discard if block cannot be processed" flag.)

2.5. Parameters and Result Fields

Various ciphersuites include several items in the security-parameters
and/or security-result fields. Which fields may appear is defined by
the particular ciphersuite description.

Each field is represented as type-length-value. Type is a single
byte indicating which item this is. Length is the count of data
bytes to follow, and is an SDNV-encoded integer. Value is the data
content of the item.

Item types are

   0: reserved

   1: initialization vector (IV)

   2: key identifier

   3: key (encoded)

   4: fragment range (offset and length as a pair of SDNVs)

   5: integrity signature

   6: reserved

   7: C block salt

   8: C block integrity check value (ICV)

   9: reserved

   10: encapsulated block

   11: reserved

   12 - 191: reserved

   192 - 250: private use

   251 - 255: reserved

The folowing descriptions apply to usage of these items for all
ciphersuites. Additional characteristics are noted in the discussion
for specific suites.

      - initialization vector(IV): random value, typically eight to
      sixteen bytes

      - key identifier: opaque identifier to indicate which key is used
      for a particular situation; format is out-of-scope for this
      specification

      - key: key to be used for a particular situation, protected as
      specified by the ciphersuite

      - fragment range: pair of SDNV values (offset then length)
      specifying the range of payload bytes to which a particular
      operation applies. This is termed "fragment range" since that is
      its typical use, even though sometimes it may describe a subset
      range that is not a fragment

      - integrity signature: result of BA or PS digest or signing
      operation

      - C block salt: an IV-like value used by certain confidentiality
      suites

      - C block integrity check value(ICV): output from certain
      confidentiality ciphersuite operations to be used at the
      destination to verify that the protected data has not been
      modified

      - encapsulated block: result of confidentiality operation on
      certain blocks, contains the ciphertext of the block and may also
      contain an integrity check value appended to the ciphertext; may
      also contain padding if required by the encryption mode; used for
      non-payload blocks only

2.6. PSB and CB combinations

   Given the above definitions, nodes are free to combine applications
   of PSB and CB in any way they wish - the correlator value allows for
   multiple applications of security services to be handled separately.

   However, there are some clear security problems that could arise when
   applying multiple services. For example, if we encrypted a payload
   but left a PSB security result containing a signature in the clear,
   payload guesses could be confirmed.

   We cannot, in general, prevent all such problems since we cannot
   assume that every ciphersuite definition takes account of every other
   ciphersuite definition. However, we can limit the potential for such
   problems by requiring that any ciphersuite which applies to one

instance of a PSB or CB, must be applied to all instances with the
same correlator.

We now list the PSB and CB combinations which we envisage as being
useful to support:

Encrypted tunnels - a single bundle may be encrypted many times
en-route to its destination. Clearly it must be decrypted an
equal number of times, but we can imagine each encryption as
representing the entry into yet another layer of tunnel. This is
supported by using multiple instances of CB, but with the payload
encrypted multiple times, "in-place". Depending upon the
ciphersuite defintion, other blocks can and should be encrypted,
as discussed above and in Section 2.4 to ensure that parameters
are protected in the case of super-encryption.

Multiple parallel authenticators - a single security source might
wish to protect the integrity of a bundle in multiple ways. This
could be required if the bundle's path is unpredictable, and if
various nodes might be involved as security destinations.
Similarly, if the security source cannot determine in advance
which algorithms to use, then using all might be reasonable. This
would result in uses of PSB which presumably all protect the
payload, and which cannot in general protect one another. Note
that this logic can also apply to a BAB, if the unpredictable
routing happens in the convergence layer, so we also envisage
support for multiple parallel uses of BAB.

Multiple sequential authenticators - if some security destination
requires assurance about the route that bundles have taken, then
it might insist that each forwarding node add its own PSB. More
likely, however would be that outbound "bastion" nodes would be
configured to sign bundles as a way of allowing the sending
"domain" to take accountability for the bundle. In this case, the
various PSBs will likely be layered, so that each protects the
earlier applications of PSB.

Authenticated and encrypted bundles - a single bundle may require
both authentication and confidentiality. Some specifications
first apply the authenticator and follow this by encrypting the
payload and authenticator. As noted previously in the case where
the authenticator is a signature, there are security reasons for
this ordering. (See the CB-RSA-AES128-PAYLOAD-PSB-CB ciphersuite
defined later in Section 4.3.) Others apply the authenticator
after encryption, that is, to the ciphertext. This ordering is
generally recommended and minimizes attacks which, in some cases,
can lead to recovery of the encryption key.

There are no doubt other valid ways to combine PSB and CB instances, but these are the "core" set supported in this specification. Having said that, as will be seen, the mandatory ciphersuites defined here are quite specific and restrictive in terms of limiting the flexibility offered by the correlator mechanism. This is primarily in order to keep this specification as simple as possible, while at the same time supporting the above scenarios.

[3](#). Security Processing

This section describes the security aspects of bundle processing.

[3.1](#). Nodes as policy enforcement points

All nodes are REQUIRED to have and enforce their own configurable
security policies, whether these policies be explicit or default, as
defined in [Section 6](#).

All nodes serve as Policy Enforcement Points (PEP) insofar as they
enforce polices that may restrict the permissions of bundle nodes to
inject traffic into the network. Policies may apply to traffic
originating at the current node, traffic terminating at the current
node and traffic to be forwarded by the current node to other nodes.
If a particular transmission request, originating either locally or
remotely, satisfies the node's policy or policies and is therefore
accepted, then an outbound bundle can be created and dispatched. If
not, then in its role as a PEP, the node will not create or forward a
bundle. Error handling for such cases is currently considered out of
scope of this document.[[Comment.5](#)]

Policy enforcing code MAY override all other processing steps
described here and elsewhere in this document. For example, it is
valid to implement a node which always attempts to attach a PSB.
Similarly it is also valid to implement a node which always rejects
all requests which imply the use of a PSB.

Nodes MUST consult their security policy to determine the criteria
that a received bundle ought to meet before it will be forwarded.
These criteria MUST include a determination of whether or not the
received bundle must include a valid BAB, PSB or CB. If the bundle
does not meet the node's policy criteria, then the bundle MUST be
discarded and processed no further; in this case, a bundle status
report indicating the failure MAY be generated.[[Comment.6](#)]

The node's policy MAY call for the node to add or subtract some
security blocks, for example, requiring the node attempt to encrypt
(parts of) the bundle for some security-destination, or requiring
that the node add a PSB. If the node's policy requires a BAB to be
added to the bundle, it MUST be added last so that the calculation of
its security result may take into consideration the values of all
other blocks in the bundle.

[3.2](#). Processing order of security blocks

The processing order of security actions for a bundle is critically
important for the actions to complete successfully. In general, the

actions performed at the originating node must be executed in the
reverse sequence at the destination. There are variations and
exceptions, and these are noted below.

The sequence is maintained in the ordering of security blocks in the
bundle. It is for this reason that blocks may not be rearranged at
forwarding nodes, whether they support the security protocols or not.
The only blocks that participate in this ordering are the primary and
payload blocks, and the security blocks themselves. All other
extension blocks are ignored for purposes of determining the
processing order.

The security blocks are added to and removed from a bundle in a last-
in-first-out (LIFO) manner, with the top of the stack immediately
after the primary block. A newly-created bundle has just the primary
and payload blocks, and the stack is empty. As security actions are
requested for the bundle, security blocks are pushed onto the stack
immediately after the primary block. The early actions have security
blocks close to the payload, later actions have blocks nearer to the
primary block. The actions deal with only those blocks in the bundle
at the time so, for example, the first to be added processes only the
payload and primary blocks, the next might process the first if it
chooses and the payload and primary, and so on. The last block to be
added can process all the blocks.

When the bundle is received, this process is reversed and security
processing begins at the top of the stack, immediately after the
primary block. The security actions are performed and the block is
popped from the stack. Processing continues with the next security
block until finally only the payload and primary blocks remain.

The simplicity of this description is undermined by various real-
world requirements. Nonetheless it serves as a helpful initial
framework for understanding the bundle security process.

The first issue is a very common one and easy to handle. The bundle
may be sent indirectly to its destination, requiring several
forwarding hops to finally arrive there. Security processing happens
at each node, assuming that the node supports bundle security. For
the following discussion, we assume that a bundle is created and that
confidentiality, then payload security and finally bundle
authentication are applied to it. The block sequence would therefore
be primary-BA-PS-C-payload. Traveling from source to destination
requires going through one intermediate node, so the trip consists of
two hops.

When the bundle is received at the intermediate node, the receive
processing validates the BA block and pops it from the stack.

However the PS and C blocks have the final destination as their
security destination, so these can't be processed and removed. The
intermediate node then begins the send process with the four
remaining blocks in the bundle. The outbound processing adds any
security blocks required by local policy, and these are pushed on the
stack immediately after the primary block, ahead of the PS block. In
this example, the intermediate node adds a PS block as a signature
that the bundle has passed through the node.

The receive processing at the destination first handles the
intermediate node's PS block and pops it, next is the originator's PS
block, also popped, and finally the originator's confidentiality
block which allows the payload to be decrypted and the bundle handled
for delivery.

This simple scheme can easily be extended to very complex networks.
It cannot deal with security zones that overlap partially but not
completely and these are discussed further below Section 3.3.
Administrators SHOULD NOT configure security-sources and security-
destinations in a network such that overlapping security zones are
created.

The second issue relates to the reversibility of certain security
process actions. In general, the actions fall into two categories:
those which do not affect other parts of the bundle, and those which
are fully reversible. Creating a bundle signature, for example, does
not change the bundle content except for the result. The encryption
performed as part of the confidentiality processing does change the
bundle, but the reverse processing at the destination restores the
original content.

The third category is the one where the bundle content has changed
slightly and in a non-destructive way, but there is no mechanism to
reverse the change. The simplest example is the addition of an EID-
reference to a security block. The addition of the reference causes
the text to be added to the bundle's dictionary. The text may be
used also by other references so removal of the block and this
specific eid-reference does not cause removal of the text from the
dictionary. This shortcoming is of no impact to the "sequential" or
"wrapping" security schemes described above, but does cause failures
with "parallel" authentication mechanisms. Solutions for this
problem are implementation-specific and typically involve multi-pass
processing such that blocks are added at one stage and the security
results calculated at a later stage of the overall process.

Certain ciphersuites have sequence requirements for their correct
operation, most notably the BundleAuthentication ciphersuites.
Processing for BundleAuthentication is required to happen after all

other sending operations, and prior to any receive operations at the
next hop node. It follows therefore that BA blocks must always be
pushed onto the stack after all others.

Although we describe the security block list as a stack, there are
some blocks which are placed after the payload and therefore are not
part of the stack. The BundleAuthentication ciphersuite #1 ("BA1")
requires a second, correlated block to contain the security-result
and this block is placed after the payload, usually as the last block
in the bundle. We can apply the stack rules even to these blocks by
specifying that they be added to the end of the bundle at the same
time that their "owner" or "parent" block is pushed on the stack. In
fact, they form a stack beginning at the payload but growing in the
other direction. Also, not all blocks in the main stack have a
corresponding entry in the trailing stack. The only blocks which
MUST follow the payload are those mandated by ciphersuites as
correlated blocks for holding a security-result. No other blocks are
required to follow the payload block and it is RECOMMENDED that they
NOT do so.
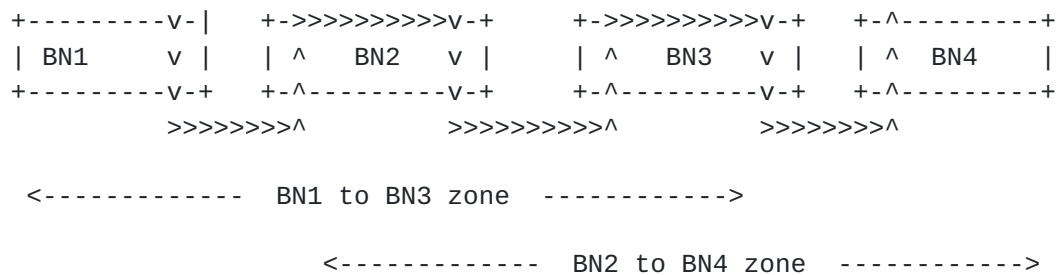
[3.3](). Security Zones

Each security block has a security zone, as described in the
discussion for Figure 1, and the zones for various blocks are often
different.

BA blocks are always for a single hop and these restricted zones
never cause conflict.

The zones for PS and C blocks are often from bundle source to bundle
destination, to provide end-to-end protection. A bundle-source-to-
bundle-destination zone likewise never causes a problem.

Another common scenario is for gateway-to-gateway protection of
traffic between two sub-networks.

Looking at Figure 1 and the simplified version shown in Figure 4, we
can regard BN2 and BN3 as gateways connecting the two subnetworks
labeled "An Internet". As long as they provide security for the BN2-
BN3 zone, all is well. Problems begin, for example, when BN2 adds
blocks with BN4 as the security-destination, and originating node BN1
has created blocks with BN3 as security-destination. We now have two
zones and neither is a subset of the other.

```
+---------v-|    +->>>>>>>>>>v-+      +->>>>>>>>>>v-+    +-^---------+
| BN1     v |    | ^   BN2   v |      | ^   BN3   v |    | ^  BN4    |
+---------v-+    +-^---------v-+      +-^---------v-+    +-^---------+
        >>>>>>>>^          >>>>>>>>>>^          >>>>>>>>^

 <------------- BN1 to BN3 zone  ------------>

                  <------------- BN2 to BN4 zone  ------------>
```

Overlapping security zones

                                Figure 4

Consider the case where the security concern is for data integrity,
so the blocks are PayloadSecurity. BN1 creates one ("PSa") along
with the new bundle, and BN2 pushes its own PS block "PSb" on the
stack, with security-destination BN4. When this bundle arrives at
BN3, the bundle blocks are

primary - PSb - PSa - payload

Block PSb is not destined for this node BN3 so must be forwarded.
This is the security-destination for block PSa so, after validation,
it should be removed from the bundle. But that will invalidate the
PSb signature when the block is checked at the final destination.
The PSb signature includes the primary block, PSb itself, PSa and the
payload block, so PSa MUST remain in the bundle. This is why
security blocks are treated as a stack and add/remove operations are
permitted only at the top-of-stack.

The situation would be worse if the security concern is
confidentiality, so C blocks are employed, such as the
confidentiality ciphersuite #3 ("C3") described in Section 4.3. In
this scenario, BN1 would encrypt the bundle with BN3 as security-
destination, BN2 would super-encrypt the payload and encapsulate the
C3 block for security-destination BN4. BN3 forwards all the blocks
without change. BN4 decrypts the payload from its super-encryption
and decapsulates the C3 block, only to find that it should have been
processed earlier. Assuming that BN4 has no access to BN3's key
store, BN4 has no way to decrypt the bundle and recover the original
content.

3.4. Canonicalisation of bundles

In order to verify a signature or MAC on a bundle the exact same
bits, in the exact same order, must be input to the calculation upon
verification as were input upon initial computation of the original
signature or MAC value. Consequently, a node MUST NOT change the

encoding of any URI in the dictionary field, e.g., changing the DNS
part of some HTTP URL from lower case to upper case. Because bundles
may be modified while in transit (either correctly or due to
implementation errors), a canonical form of any given bundle (that
contains a BAB or PSB) must be defined.

This section defines two bundle canonicalisation algorithms which can
be used by various ciphersuites.

3.4.1. Strict canonicalisation

The first algorithm that can be used permits no changes at all to the
bundle between the security-source and the security-destination. It
is mainly intended for use in BAB ciphersuites. This algorithm
conceptually catenates all blocks in the order presented, but omits
all security result data fields in blocks of this ciphersuite type.
That is, when a BA ciphersuite specifies this algorithm then we omit
all BAB security results for all BA ciphersuites, when a PSB
ciphersuite specifies this algorithm then we omit all PSB security
results for all PS ciphersuites. All security result length fields
are included, even though their corresponding security result data
fields are omitted.

Notes:

   - In the above we specify that security result data is omitted.
   This means that no bytes of the security result data are input.
   We do not set the security result length to zero. Rather, we
   assume that the security result length will be known to the module
   that implements the ciphersuite before the security result is
   calculated, and require that this value be in the security result
   length field even though the security result data itself will be
   omitted.

   - The 'res' bit of the ciphersuite ID, which indicates whether or
   not the security result length and security result data field are
   present, is part of the canonical form.

   -The value of the block data length field, which indicates the
   length of the block, is also part of the canonical form. Its
   value indicates the length of the entire bundle when the bundle
   includes the security result data field.

   -BABs are always added to bundles after PSBs, so when a PSB
   ciphersuite specifies this strict canonicalisation algorithm and
   the PSB is received with a bundle that also includes one or more
   BABs, application of strict canonicalisation as part of the PSB
   security result verification process requires that all BABs in the
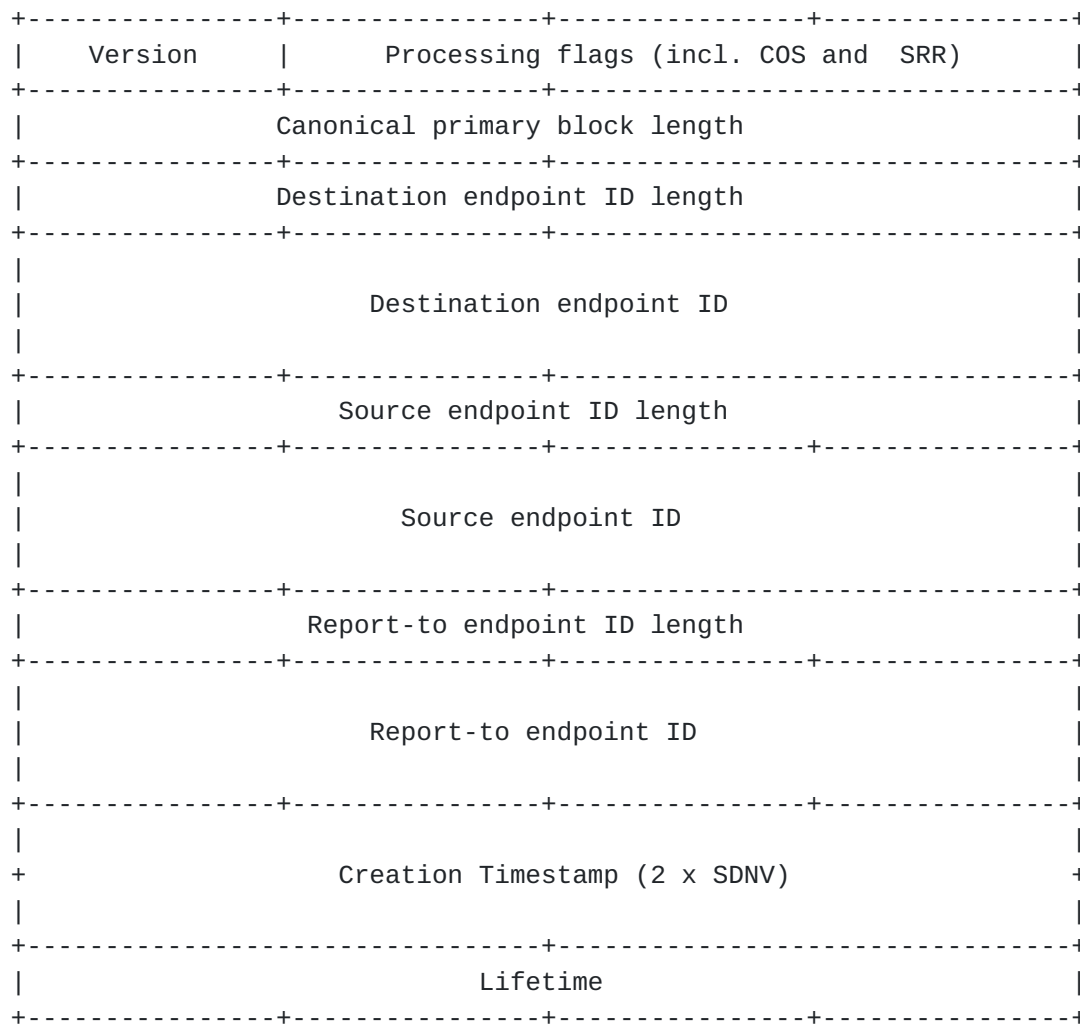
bundle be ignored entirely.

3.4.2. Mutable canonicalisation

This algorithm is intended to protect parts of the bundle which
should not be changed in-transit. Hence it omits the mutable parts
of the bundle.

The basic approach is to define a canonical form of the primary block
and catenate it with the security and payload blocks in the order
that they will be transmitted. This algorithm ignores all other
blocks because it cannot be determined whether or not they will
change as the bundle transits the network.

Many fields in various blocks are stored as variable-length SDNVs.
These are canonicalized in unpacked form, as eight-byte fixed-width
fields in network byte order. The size of eight bytes is chosen
because implementations may handle larger values as invalid, as noted
in [2].

The canonical form of the primary block is shown in Figure 5.
Essentially, it de-references the dictionary block, adjusts lengths
where necessary and ignores flags that may change in transit.

```
+---------------+---------------+-------------------------------+
|    Version    |     Processing flags (incl. COS and  SRR)    |
+---------------+---------------+-------------------------------+
|              Canonical primary block length                  |
+---------------+---------------+-------------------------------+
|              Destination endpoint ID length                  |
+---------------+---------------+-------------------------------+
|                                                              |
|                                                              |
|                 Destination endpoint ID                      |
|                                                              |
|                                                              |
+---------------+---------------+-------------------------------+
|                Source endpoint ID length                     |
+---------------+---------------+---------------+---------------+
|                                                              |
|                                                              |
|                   Source endpoint ID                         |
|                                                              |
|                                                              |
+---------------+---------------+-------------------------------+
|               Report-to endpoint ID length                   |
+---------------+---------------+---------------+---------------+
|                                                              |
|                                                              |
|                  Report-to endpoint ID                       |
|                                                              |
|                                                              |
+---------------+---------------+---------------+---------------+
|                                                              |
+                 Creation Timestamp (2 x SDNV)                +
|                                                              |
+-------------------------------+-------------------------------+
|                           Lifetime                           |
+---------------+---------------+---------------+---------------+
```

   The canonical form of the primary bundle block.

                                Figure 5

   The fields shown in Figure 5 are:

      Version is the single-byte value in the primary block.

      Processing flags in the primary block is an SDNV, and includes the
      class-of-service (COS) and status report request (SRR) fields.
      For purposes of canonicalization, the SDNV is unpacked into a
      fixed-width field and some bits are masked out. The unpacked
      field is ANDed with mask 0x0000 0000 003E 031F to set to zero all
      reserved bits and the "bundle is a fragment" bit.

      Length - a four-byte value containing the length (in bytes) of
      this structure, in network byte order.[Comment.7]

Destination endpoint ID length and value - are the length (as a four byte value in network byte order) and value of the destination endpoint ID from the primary bundle block. The URI is simply copied from the relevant part(s) of the dictionary block and is not itself canonicalised. Although the dictionary entries contain null-terminators, the null-terminators are not included in the length or the canonicalization.

Source endpoint ID length and value are handled similarly to the destination.

Report-to endpoint ID length and value are handled similarly to the destination.

Creation time (2 x SDNV) and Lifetime (SDNV) are simply copied from the primary block, with the SDNV values being represented as eight-byte unpacked values.

Fragment offset and Total application data unit length are ignored, as is the case for the "bundle is a fragment" bit mentioned above. If the payload data to be canonicalized is less than the complete, original bundle payload, the offset and length are specified in the security-parameters.

For non-primary blocks being included in the canonicalization, the block processing flags value used for canonicalization is the unpacked SDNV value with reserved and mutable bits masked to zero. The unpacked value is ANDed with mask 0x0000 0000 0000 0057 to zero reserved bits and the "last block" flag. The "last block" flag is ignored because BABs and other security blocks may be added for some parts of the journey but not others so the setting of this bit might change from hop to hop.

Endpoint ID references in security blocks are canonicalized using the de-referenced text form in place of the reference pair. The reference count is not included, nor is the length of the endpoint ID text.

The block-length is canonicalized as an eight-byte unpacked value in network byte order. If the payload data to be canonicalized is less than the complete, original bundle payload, this field contain the size of the data being canonicalized (the "effective block") rather that the actual size of the block.

Payload blocks are generally cononicalized as-is with the exception that in some instances only a portion of the payload data is to be protected. In such a case, only those bytes are included in the canonical form, and additional ciphersuite parameters are required to

specify which part of the payload is protected, as discussed further
below.

Security blocks are handled likewise, except that the ciphersuite
will likely specify that the "current" security block security result
field not be considered part of the canonical form. This differs
from the strict canonicalisation case since we might use the mutable
canonicalisation algorithm to handle sequential signatures such that
signatures cover earlier ones.

Notes:

   - The canonical form of the bundle is not transmitted. It is
   simply an artifact used as input to digesting.

   - We omit the reserved flags because we cannot determine if they
   will change in transit. The masks specified above may have to be
   revised if additional flags are defined and they need to be
   protected.

   - Our URI encoding does not preserve the "null-termination"
   convention from the dictionary field, nor do we separate the
   scheme and ssp as is done there.

   - The URI encoding will cause errors if any node rewrites the
   dictionary content (e.g. changing the DNS part of an HTTP URL from
   lower-case to upper case). This could happen transparently when a
   bundle is synched to disk using one set of software and then read
   from disk and forwarded by a second set of software. Because
   there are no general rules for canonicalising URIs (or IRIs), this
   problem may be an unavoidable source of integrity failures.

   - All SDNV fields here are canonicalized as eight-byte unpacked
   values in network byte order. Length fields are canonicalized as
   four-byte values in network byte order. Encoding does not need
   optimization since the values are never sent over the network.

   If a bundle is fragmented before the PSB is applied then the PSB
   applies to a fragment and not the entire bundle. However, the
   protected fragment could be subsequently further fragmented, which
   would leave the verifier unable to know which bytes were protected
   by the PSB. Even in the absence of frgmentation the same
   situation applies if the ciphersuite is defined to allow
   protection of less than the entire, original bundle payload.

   For this reason, PSB ciphersuites which support applying a PSB to
   less than the complete, original bundle payload MUST specify, as
   part of the ciphersuite parameters, which bytes of the bundle

payload are protected. When verification occurs, only the
specified range of the payload bytes are input to PSB
verification. It is valid for a ciphersuite to be specified so as
to only apply to entire bundles and not to fragments. A
ciphersuite may be specified to apply to only a portion of the
payload, regardless of whether the payload is a fragment or the
complete original bundle payload.

The same fragmentation issue applies equally to CB ciphersuites.
Ciphersuites which support applying confidentiality to fragments
MUST specify, as part of the ciphersuite parameters, which bytes
of the bundle payload are protected. When decrypting a fragment,
only the specified bytes are processed. It is also valid for a
confidentiality ciphersuite to be specified so as to only apply to
entire bundles and not to fragments.

3.5. Endpoint ID confidentiality

Every bundle MUST contain a primary block that cannot be encrypted
which also contains the source and destinations endpoint IDs, and
others. If endpoint ID confidentiality is required, then a fake
primary block with false values for these fields must be constructed
and with a new block type to contain the actual values. There is
presently no mechanism defined that supports such a "fake block" or
possibly "fake dictionary". Bundle-in-bundle encapsulation may solve
this problem in some instances.

Similarly, confidentiality requirements may also apply to other parts
of the primary block (e.g. the current-custodian) and that is
supported in the same manner.

3.6. Bundles received from other nodes

Nodes implementing this specification SHALL consult their security
policy to determine whether or not a received bundle is required by
policy to include a BAB. If the bundle is not required to have a
BAB, then BAB processing on the received bundle is complete and the
bundle is ready to be further processed for CB/PSB handling or
delivery or forwarding.

If the bundle is required to have a BAB but does not, then the bundle
MUST be discarded and processed no further. If the bundle is
required to have a BAB but all of its BABs identify a different node
other than the receiving node as the BAB security destination, then
the bundle MUST be discarded and processed no further.

If the bundle has one or more BABs that identify the receiving node
as the BAB security destination, or for which there is no security

destination, then the value in the security result field(s) of the
BAB(s) MUST be verified according to the ciphersuite specification.
If for all such BABs in the bundle either the BAB security source
cannot be determined or the security result value check fails, the
bundle has failed to authenticate and the bundle MUST be discarded
and processed no further. Otherwise, if any of the BABs present
verify, the bundle is ready for further processing as determined by
extension blocks and/or policy.

BABs received in a bundle MUST be stripped before the bundle is
forwarded. New BABs MAY be added as required by policy. This may
require correcting the "last block" field of the to-be-forwarded
bundle.

Further processing of the bundle must take place in the order
indicated by the various blocks from the primary block to the payload
block, except as defined by an applicable specification.

If the bundle has a CB and the receiving node is the CB destination
for the bundle (either because the node is listed as the bundle's CB-
dest or because the node is listed as the bundle's destination and
there is no CB-dest), the node MUST decrypt the relevant parts of the
bundle in accordce with the ciphersuite specification. The CB SHALL
be deleted. If the relevant parts of the bundle cannot be decrypted
(i.e., the decryption key cannot be deduced or decryption fails),
then the bundle MUST be discarded and processed no further; in this
case a bundle deletion status report (see the Bundle Protocol [2])
indicating the decryption failure MAY be generated. If the CB
security result included the ciphertext of a block other than the
payload block, the recovered plaintext block MUST be placed in the
bundle at the location from which the CB was deleted.[Comment.8]

If the bundle has one or more PSBs for which the receiving node is
the bundle's PSB destination (either because the node is listed in
the bundle's PSB-dest or because the node is listed as the bundle's
destination and there is no PSB-dest), the node MUST verify the value
in the PSB security result field(s) in accordance with the
ciphersuite specification. If all the checks fail, the bundle has
failed to authenticate and the bundle SHALL be processed according to
the security policy. A bundle status report indicating the failure
MAY be generated. Otherwise, if the PSB verifies, the bundle is
ready to be processed for either delivery or forwarding. Before
forwarding the bundle, the node SHOULD remove the PSB from the
bundle, subject to the requirements of Section 3.2, unless it is
likely that some downstream node will also be able to verify the PSB.

If the bundle has a PSB and the receiving node is not the bundle's
PSB-dest the receiving node MAY attempt to verify the value in the

security result field. If it is able to check and the check fails,
the node SHALL discard the bundle and it MAY send a bundle status
report indicating the failure.[Comment.9][Comment.10]

 3.7.   The At-Most-Once-Delivery Option

   An application may request (in an implementation specific manner)
   that a node be registered as a member of an endpoint and that
   received bundles destined for that endpoint be delivered to that
   application.

   An option for use in such cases is known as "at-most-once-delivery".
   If this option is chosen, the application indicates that it wants the
   node to check for duplicate bundles, discard duplicates, and deliver
   at most one copy of each received bundle to the application. If this
   option is not chosen, the application indicates that it wants the
   node to deliver all received bundle copies to the application. If
   this option is chosen, the node SHALL deliver at most one copy of
   each received bundle to the application. If the option is not
   chosen, the node SHOULD, subject to policy, deliver all bundles.

   To enforce this the node MUST look at the source/timestamp pair value
   of each complete (reassembled, if necessary) bundle received and
   determine if this pair, which uniquely identifies a bundle, has been
   previously received. If it has, then the bundle is a duplicate. If
   it has not, then the bundle is not a duplicate. The source/timestamp
   pair SHALL be added to the list of pair values already received by
   that node.

   Each node implementation may decide how long to maintain a table of
   pair value state.

   Additional discussion relevant to at-most-delivery is in the DTN
   Retransmission Block specification [10].

3.8. Bundle Fragmentation and Reassembly

   If it is necessary for a node to fragment a bundle and security
   services have been applied to that bundle, the fragmentation rules
   described in [2] MUST be followed. As defined there and repeated
   here for completeness, only the payload may be fragmented; security
   blocks, like all extension blocks, can never be fragmented. In
   addition, the following security-specific processing is REQUIRED:

   The security policy requirements for a bundle must be applied
   individually to all the bundles resulting from a fragmentation event.

   If the original bundle contained a PSB, then each of the PSB

instances MUST be included in some fragment.

If the original bundle contained one or more CBs, then any CB
instances containing a BEK MUST have the "replicate in every
fragment" flag set, and thereby be replicated in every fragment.
This is to ensure that the canonical block-sequence can be recovered
during reassembly.

If the original bundle contained one or more correlated CBs not
containing a BEK, then each of these MUST be included in some
fragment, but SHOULD NOT be sent more than once. They MUST be placed
in a fragment in accordance with the fragmentation rules described in
[2].

Note: various fragments may have additional security blocks added at
this or later stages and it is possible that correlators may collide.
In order to facilitate uniqueness, ciphersuites SHOULD include the
fragment-offset of the fragment as a high-order component of the
correlator.

3.9. Reactive fragmentation

When a partial bundle has been received, the receiving node SHALL
consult its security policy to determine if it may fragment the
bundle, converting the received portion into a bundle fragment for
further forwarding. Whether or not reactive fragmentation is
permitted SHALL depend on the security policy and the ciphersuite
used to calculate the BAB authentication information, if required.
(Some BAB ciphersuites, i.e., the mandatory BAB-HMAC ciphersuite
defined in Section 4.1, do not accommodate reactive fragmentation
because the security result in the BAB requires that the entire
bundle be signed. It is conceivable, however, that a BAB ciphersuite
could be defined such that multiple security results are calculated,
each on a different segment of a bundle, and that these security
results could be interspersed between bundle payload segments such
that reactive fragmentation could be accommodated.)

If the bundle is reactively fragmented by the intermediate receiver
and the BAB-ciphersuite is of an appropriate type (e.g. with multiple
security results embedded in the payload), the bundle MUST be
fragmented immediately after the last security result value in the
partial payload that is received. Any data received after the last
security result value MUST be dropped.

If a partial bundle is received at the intermediate receiver and is
reactively fragmented and forwarded, only the part of the bundle that
was not received MUST be retransmitted, though more of the bundle MAY
be retransmitted. Before retransmitting a portion of the bundle, it

   SHALL be changed into a fragment and, if the original bundle included
   a BAB, the fragmented bundle MUST also, and its BAB SHALL be
   recalculated.

   This specification does not currently define any ciphersuite which
   can handle this reactive fragmentation case.

   An interesting possibility is a ciphersuite definition such that the
   transmission of a follow-up fragment would be accompanied by the
   signature for the payload up to the restart point.

4. **Mandatory Ciphersuites**

   This section defines the mandatory ciphersuites for this
   specification. There is currently one mandatory ciphersuite for use
   with each security block type (BAB, PSB and CB). The BAB ciphersuite
   is based on shared secrets using HMAC. The PSB ciphersuite is based
   on digital signatures using RSA with SHA256. The CB ciphersuite is
   based on using RSA for key transport and AES for bulk encryption.

4.1. **BAB-HMAC**

   The BAB-HMAC ciphersuite has ciphersuite ID value 0x001.

   Security parameters are optional with this scheme, but if used then
   the only field that can be present is the key identifier (see
   Section 2.5). The exact type of key identifier to be used is an
   implementation issue. In the absence of a key identifier the
   receiver is expected to be able to find the correct key based on the
   sending identity. The sending identity may be known from the
   security-source field or the content of a previous-hop block in the
   bundle. It may also be determined using implementation-specific
   means such as the convergence layer.

   BAB-HMAC uses the strict canonicalisation algorithm in Section 3.4.1.

   Strict canonicalization supports digesting of a fragment-bundle. It
   does not permit the digesting of only a subset of the payload, but
   only the complete contents of the payload of the current bundle,
   which might be a fragment. The "fragment range" item for security-
   parameters is not used to indicate a fragment, as this information is
   digested within the primary block.

   The variant of HMAC to be used is HMAC-SHA1 as defined in RFC 2104
   [3].[Comment.11]

   This ciphersuite requires the use of two related instances of the
   BAB. It involves placing the first BAB instance (as defined in
   Section 2.2) just after the primary block. The second (correlated)
   instance of the BAB MUST be placed after all other blocks (except
   possibly other BAB blocks) in the bundle.

   This means that normally, the BAB will be the second and last blocks
   of the bundle. If a forwarder wishes to apply more than one
   correlated BAB pair, then this can be done. There is no requirement
   that each application "wrap" the others, but the forwarder MUST
   insert all the "up front" BABs, and their "at back" "partners"
   (without any security result), before canonicalising.

Inserting more than one correlated BAB pair would be useful if the
bundle could be routed to more than one potential "next-hop" or if
both an old or a new key were valid at sending time, with no
certainty about the situation that will obtain at reception time.

The security result is the output of the HMAC-SHA1 calculation with
input being the result of running the entire bundle through the
strict canonicalisation algorithm. Both required BAB instances MUST
be included in the bundle before canonicalisation.

4.2. PSB-RSA-SHA256

The PSB-RSA-SHA256 ciphersuite has ciphersuite ID value 0x002.

If the bundle being signed has been fragmented before signing, then
we have to specify which bytes were signed in case the signed bundle
is subsequently fragmented for a second time. If the bundle is a
fragment, then the ciphersuite parameters MUST include a fragment-
range field, as described in Section 2.5, specifying the offset and
length of the signed fragment. If the entire bundle is signed then
these numbers MUST be omitted.

The ciphersuite parameters field MAY also contain a key identifier.
The exact type of key identifier to be used is an implementation
issue. In the absence of a key identifier, the verifier of the PSB
is expected to be able to use the security source (if supplied) or
else the bundle source (if no security source is present) in order to
determine the correct public key to use for PSB verification.

PSB-RSA-SHA256 uses the mutable canonicalisation algorithm
Section 3.4.2, with the security-result data field for only the
"current" block being excluded from the canonical form. The
resulting canonical form of the bundle is the input to the signing
process. This ciphersuite requires the use of a single instance of
the PSB.

RSA is used with SHA256 as specified for the sha256WithRSAEncryption
PKCSv1.5 signature scheme in RFC 4055 [4]. The output of the signing
process is the security result item for the PSB.

"Commensurate strength" cryptography is generally held to be a good
idea. A combination of RSA with SHA256 is reckoned to require a 3076
bit RSA key according to this logic. Few implementations will choose
this length by default (and probably some just won't support such
long keys). Since this is an experimental protocol, we expect that
1024 or 2048 bit RSA keys will be used in many cases, and that that
will be fine since we also expect that the hash function "issues"
will be resolved before any standard would be derived from this

protocol.[Comment.12]

4.3.   CB-RSA-AES128-PAYLOAD-PSB-CB
   [Comment.13]

   The CB-RSA-AES128-PAYLOAD-PSB-CB ciphersuite has ciphersuite ID value
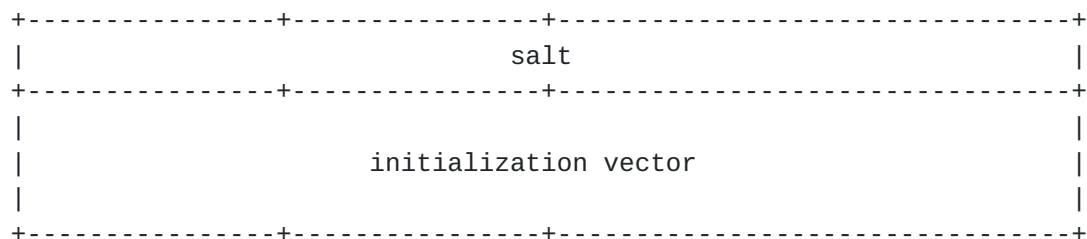   0x003.

   This scheme encrypts PSBs, CBs for this suite #3 ("C3"), and the
   payload. CBs for other ciphersuites are not encrypted. The key size
   for this ciphersuite is 128 bits.

   Encryption is done using the AES algorithm in Galois/Counter Mode
   (GCM) in a manner generally similar to that used for IPSec ESP, as
   described in RFC 4106 [5], from which parts of the following
   description are borrowed.

   The choice of GCM avoids expansion of the payload, which causes
   problems with fragmentation/reassembly and custody transfer. GCM
   also includes authentication, essential in preventing attacks that
   can alter the decrypted plaintext or even recover the encryption key.

   GCM is a block cipher mode of operation providing both
   confidentiality and data origin authentication. The GCM
   authenticated encryption operation has four inputs: a secret key, an
   initialization vector (IV), a plaintext, and an input for additional
   authenticated data (AAD) which is not used here. It has two outputs,
   a ciphertext whose length is identical to the plaintext, and an
   authentication tag, also known as the Integrity Check Value (ICV).

   For consistency with the description in [5], we refer to the GCM IV
   as a nonce. The same key and nonce combination MUST NOT be used more
   than once. The nonce has the following layout

```
+----------------+----------------+-------------------------------+
|                             salt                                |
+----------------+----------------+-------------------------------+
|                                                                 |
|                      initialization vector                      |
|                                                                 |
+----------------+----------------+-------------------------------+
```

   Nonce format.

                              Figure 6

   The salt field is a four-octet value, usually chosen at random. It

MUST be the same for all CBs which have the same correlator value.
The salt need not be kept secret.

The initialization vector (IV) is an eight-octet value, usually
chosen at random. It MUST be different for all CBs which have the
same correlator value. The value need not be kept secret.

The key (bundle encryption key, BEK) is a sixteen-octet (128 bits)
value, usually chosen at random. The value MUST be kept secret, as
described below.

The integrity check value is a sixteen-octet value used to verify
that the protected data has not been altered. The value need not be
kept secret.

This ciphersuite requires the use of a single CB instance to deal
with payload confidentiality. If the bundle already contains PSBs or
"C3" CBs then the ciphersuite will create additional correlated
blocks to protect these PSBs and C3 CBs. These "additional" blocks
replace the original blocks on a one-for-one basis, so the number of
blocks remains unchanged. All these related blocks MUST have the
same correlator value. The term "first CB" in this section refers to
the single CB if there is only one or, if there are several, then to
the one containing the key or key-identifier. This MUST be the first
of the set.

First CB - the first CB may contain a correlator value, and may
specify security-source and/or security-destination in the eid-list.
If not specified, the bundle-source and bundle-destination
respectively are used for these values, as with other ciphersuites.
The block MUST contain security-parameters and security-result
fields. Each field may contain several items formatted as described
in [Section 2.5](#).

Security-parameters

   salt

   IV (this instance applies only to payload)

   key identifier (optional)

   fragment offset and length, if bundle is a fragment

Security-result

   key

   ICV

Subsequent CBs MUST contain a correlator value to link them to the
first CB. Security-source and security-destination are implied from
the first CB, however see the discussion below concerning eid-list
entries. They MUST contain security-parameters and security-result
fields as follows:

Security-parameters

   IV for this specific block

Security-result

   encapsulated block

The security-parameters and security-result fields in the subsequent
CBs MUST NOT contain any items other than these two. Items such as
key and salt are supplied in the first CB and MUST NOT be repeated.

Protection of the key can be done in two ways. It can be encrypted
using PKCSv1.5 rsaEncryption as specified in RFC 3370 [6] and the
result placed in the security-result field of the first CB.
Alternatively, keys can be negotiated or pre-shared and a key-
identifier for the chosen one provided in the security-parameters
field of the first CB. Format of the key-identifier and details of
sharing are out of scope here. Behavior is not specified if both a
key and a key-identifier are included.

If the bundle being encrypted is a fragment-bundle we have to specify
which bytes are encrypted in case the bundle is subsequently
fragmented again. If the bundle is a fragment the ciphersuite
parameters MUST include a fragment-range field, as described in
Section 2.5, specifying the offset and length of the encrypted
fragment. Note that this is not the same pair of fields which appear
in the primary block as "offset and length". The "length" in this
case is the length of the fragment, not the original length. If the
bundle is not a fragment then this field MUST be omitted.

The confidentiality processing for payload and other blocks is
different, mainly because the payload might be fragmented later at
some other node.

For the payload, only the bytes of the bundle payload field are
affected, being replaced by ciphertext. The salt, IV and key values
specified in the first CB are used to encrypt the payload, and the
resultant authentication tag (ICV) is placed in an ICV item in the
security-result field of that first CB. The other bytes of the

payload block, such as type, flags and length, are not modified.

For each PSB or CB to be protected, the entire original block is
encapsulated in a "replacing" CB. This replacing CB is placed in the
outgoing bundle in the same position as the original block, PSB or
CB. As mentioned above, this is one-for-one replacement and there is
no consolidation of blocks or mixing of data in any way.

The encryption process uses AES-GCM with the salt and key values from
the first CB, and an IV unique to this CB. The process creates
ciphertext for the entire original block, and an authentication tag
for validation at the security destination. For this encapsulation
process, unlike the processing of the bundle payload, the
authentication tag is appended to the ciphertext for the block and
the combination is stored into the "encapsulated block" item in
security-result.

The replacing block, of course, also has the same correlator value as
the first CB with which it is associated. It also contains the
block-specific IV in security-parameters, and the combination of
original-block-ciphertext and authentication tag, stored as an
"encapsulated block" item in security-result.

Any existing eid-list in the to-be-encapsulated original block
remains exactly as-is, and become the eid-list for the replacing
block. The encapsulation process MUST NOT replace or remove the
existing eid-list entries. This is critically important for correct
updating of entries at the security-destination.

At the security-destination, either specific destination or the
bundle destination, the processes described above are reversed. The
payload is decrypted in-place using the salt, IV and key values in
the first CB, including verification using the ICV. Each correlated
CB is also processed at the same destination, using the salt and key
values from the first CB and the block-specific IV item. The
"encapsulated block" item in security-result is decrypted, and
validated using the tag which was appended to the ciphertext of the
original block data. Assuming the validation succeeds, the resultant
plaintext, which is the entire content of the original block,
replaces the CB at the same place in the bundle. The block type
reverts to that of the original block prior to encapsulation, and the
other block-specific data fields also return to their original
values. Implementors are cautioned that this "replacement" process
requires delicate stitchery, as the eid-list contents in the
decapsulated block are invalid. As noted above, the eid-list
references in the original block were preserved in the replacing CB,
and will have been updated as necessary as the bundle has toured the
dtnet. The references from the CB MUST replace the references within

the eid-list of the newly-decapsulated block. Caveat implementor.

If the payload was fragmented after encryption then all those
fragments MUST be present and reassembled before decryption. This
process mighht be repeated several times at different destinations if
multiple fragmentation actions have occurred.

The size of the GCM counter field limits the payload size to 2^39 -
256 bytes, about half a terabyte. A future revision of this
specification will address the issue of handling payloads in excess
of this size.

5. **Key Management**

   Since key management in delay tolerant networks is still a research
   topic we cannot provide much in the way of useful key management
   here. However, solely in order to support implementation and
   testing, implementations SHOULD support:

      - Long-term pre-shared-symmetric keys for the BAB-HMAC
      ciphersuite.

      - The use of well-known RSA public keys for PSB-RSA-SHA256 and CB-
      RSA-AES128-PAYLOAD-PSB-CB ciphersuites.

   Since endpoint IDs are URIs and URIs can be placed in X.509 [7]
   public key certificates (in the subjectAltName extension)
   implementations SHOULD support this way of distributing public keys.
   Implementations SHOULD NOT be very strict in how they process X.509
   though, for example, it would probably not be correct to insist on
   Certificate Revocation List (CRL) checking in many DTN contexts.

   Other than that, key management is for future study.

6. **Default Security Policy**

   Every node serves as a Policy Enforcement Point insofar as it
   enforces some policy that controls the forwarding and delivery of
   bundles via one or more convergence layer protocol implementation.
   Consequently, every node SHALL have and operate according to its own
   configurable security policy, whether the policy be explicit or
   default. The policy SHALL specify:

      Under what conditions received bundles SHALL be forwarded.

      Under what conditions received bundles SHALL be required to
      include valid BABs.

      Under what conditions the authentication information provided in a
      bundle's BAB SHALL be deemed adequate to authenticate the bundle.

      Under what conditions received bundles SHALL be required to have
      valid PSBs and/or CBs.

      Under what conditions the authentication information provided in a
      bundle's PSB SHALL be deemed adequate to authenticate the bundle.

      Under what conditions a BAB SHALL be added to a received bundle
      before that bundle is forwarded.

      Under what conditions a PSB SHALL be added to a received bundle
      before that bundle is forwarded.

      Under what conditions a CB SHALL be added to a received bundle
      before that bundle is forwarded.

      The actions that SHALL be taken in the event that a received
      bundle does not meet the receiving node's security policy
      criteria.

   This specification does not address how security policies get
   distributed to nodes. It only REQUIRES that nodes have and enforce
   security policies. [Comment.14]

   If no security policy is specified at a given node, or if a security
   policy is only partially specified, that node's default policy
   regarding unspecified criteria SHALL consist of the following:

      Bundles that are not well-formed do not meet the security policy
      criteria.

The mandatory ciphersuites MUST be used.

All bundles received MUST have a BAB which MUST be verified to contain a valid security result. If the bundle does not have a BAB, then the bundle MUST be discarded and processed no further; a bundle status report indicating the authentication failure MAY be generated.

No received bundles SHALL be required to have a PSB; if a received bundle does have a PSB, however, the PSB can be ignored unless the receiving node is the PSB-dest, in which case the PSB MUST be verified.

No received bundles SHALL be required to have a CB; if a received bundle does have a CB, however, the CB can be ignored unless the receiving node is the CB-dest, in which case the CB MUST be processed. If processing of a CB yields a PSB, that PSB SHALL be processed by the node according to the node's security policy.

A PSB SHALL NOT be added to a bundle before sourcing or forwarding it.

A CB SHALL NOT be added to a bundle before sourcing or forwarding it.

A BAB MUST always be added to a bundle before that bundle is forwarded.

If a destination node receives a bundle that has a PSB-dest but the value in that PSB-dest is not the EID of the destination node, the bundle SHALL be delivered at that destination node.

If a received bundle does not satisfy the node's security policy for any reason, then the bundle MUST be discarded and processed no further; in this case, a bundle deletion status report (see the Bundle Protocol [2]) indicating the failure MAY be generated.

7. **Security Considerations**
    [Comment.15]

   If a BAB ciphersuite uses digital signatures but doesn't include the
   security destination (which for a BAB is the next host), then this
   allows the bundle to be sent to some node other than the intended
   adjacent node. Because the BAB will still authenticate, the
   receiving node may erroneously accept and forward the bundle. When
   asymmetric BAB ciphersuites are used, the security destination field
   SHOULD therefore be included in the BAB.

   If a bundle's PSB-dest is not the same as its destination, then some
   node other than the destination (the node identified as the PSB-dest)
   is expected to validate the PSB security result while the bundle is
   en route. However, if for some reason the PSB is not validated,
   there is no way for the destination to become aware of this.
   Typically, a PSB-dest will remove the PSB from the bundle after
   verifying the PSB and before forwarding it. However, if there is a
   possibility that the PSB will also be verified at a downstream node,
   the PSB-dest will leave the PSB in the bundle. Therefore, if a
   destination receives a bundle with a PSB that has a PSB-dest (which
   isn't the destination), this may, but does not necessarily, indicate
   a possible problem.

   If a bundle is fragmented after being forwarded by its PSB-source but
   before being received by its PSB-dest, the payload in the bundle MUST
   be reassembled before validating the PSB security result in order for
   the security result to validate correctly. Therefore, if the PSB-
   dest is not capable of performing payload reassembly, its utility as
   a PSB-dest will be limited to validating only those bundles that have
   not been fragmented since being forwarded from the PSB-source.
   Similarly, if a bundle is fragmented after being forwarded by its
   PSB-source but before being received by its PSB-dest, all fragments
   MUST be received at that PSB-dest in order for the bundle payload to
   be able to be reassembled. If not all fragments are received at the
   PSB-dest node, the bundle will not be able to be authenticated, and
   will therefore never be forwarded by this PSB-dest node.

   Specification of a security-destination other than the bundle
   destination creates a routing requirement that the bundle somehow be
   directed to the security-destination node on its way to the final
   destination. This requirement is presently private to the
   ciphersuite, since routing nodes are not required to implement
   security processing.

8. IANA Considerations

None at this time. If the bundle protocol becomes a standards track
protocol, then we may want to consider having IANA establish a
register of block types, and in particular for this specification a
separate register of ciphersuite specifications.

9. References

9.1. Normative References

[1]   Bradner, S. and J. Reynolds, "Key words for use in RFCs to
        Indicate Requirement Levels", RFC 2119, October 1997.

[2]   Scott, K. and S. Burleigh, "Bundle Protocol Specification",
        draft-irtf-dtnrg-bundle-spec-10.txt, work-in-progress,
        July 2007.

[3]   Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing
        for Message Authentication", RFC 2104, February 1997.

[4]   Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms
        and Identifiers for RSA Cryptography for use in the Internet
        X.509 Public Key Infrastructure Certificate and Certificate
        Revocation List (CRL) Profile", RFC 4055, June 2005.

[5]   Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM)
        in IPsec Encapsulating Security Payload (ESP)", RFC 4106,
        June 2005.

[6]   Housley, R., "Cryptographic Message Syntax (CMS) Algorithms",
        RFC 3370, August 2002.

[7]   Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509
        Public Key Infrastructure Certificate and Certificate
        Revocation List (CRL) Profile", RFC 3280, April 2002.

9.2. Informative References

[8]   Farrell, S., Symington, S., and H. Weiss, "Delay-Tolerant
        Network Security Overview",
        draft-irtf-dtnrg-sec-overview-03.txt, work-in-progress,
        April 2007.

[9]   Cerf, V., Burleigh, S., Durst, R., Fall, K., Hooke, A., Scott,
        K., Torgerson, L., and H. Weiss, "Delay-Tolerant Network
        Architecture", RFC 4838, April 2007.

[10]   Symington, S., "Delay-Tolerant Network Retransmission Block",
        draft-irtf-dtnrg-bundle-retrans-00.txt, work-in-progress,
        April 2007.

Editorial Comments

   [Comment.1]   Stephen: I guess there could be some weird corner case
                    where a CB ciphersuite using counter-mode would allow
                    fragments to be individually decrypted, and in that
                    case, we might want to set replication for each
                    fragment. So we can't fully rule out setting that flag
                    for all PSB/CB.

   [Comment.2]   plovell: in fact, we do need it set for C blocks (first
                    only if a correlated set) in order to recover canonical
                    processing order after fragmentation/reassembly

   [Comment.3]   Stephen: This to be revisited!

   [Comment.4]   Note: there is no ciphersuite definition which supports
                    this "parallel" encryption scheme -- needs thought. All
                    definitions so far are for sequential encryptors, and
                    we don't have a way to indicate "parallel". Maybe it's
                    just that the CB-dest is different? I hate to think
                    what the API would look like :)

   [Comment.5]   Stephen: Do we need to specify error handling for the
                    case where a node drops a bundle for policy reasons?
                    Does/can it signal back to the source that its done so?

   [Comment.6]   Howie: The security policy database will need to be
                    discussed somewhere. Does it belong in this document,
                    the bundle protocol spec., both, some other document?

   [Comment.7]   Editors: Check that mask value at the very last moment
                    (incl. during auth-48) to be sure its (still) correct.

   [Comment.8]   plovell: do we generate a report even if the failure
                    was a security failure? That's often regarded as poor
                    practice.

   [Comment.9]   plovell: should this be "SHALL discard" or "SHALL
                    delete" ?

   [Comment.10]   plovell: do we generate a report even if the failure
                    was a security failure? That's often regarded as poor
                    practice.

   [Comment.11]   Editors: At the moment there appears to be no security
                    reason to move away from HMAC-SHA1 since the HMAC
                    construct is not as far as we know affected by
                    collisions in the underlying digest algorithm (which

are nearly practically computable for SHA-1).
Nevertheless, since we use SHA-256 in the signature
ciphersuite (since collisions do matter there), it may
be desirable to move to HMAC-SHA-256 as specified in
RFC 4321. So if you're writing code based on this...be
warned!

[Comment.12] Editors: There are currently unresolved "issues" with
digest algorithms which might cause a change here prior
to, but more likely, after, an RFC has issued. So
expect change!

[Comment.13] Editors: This section has progressed to Woodman status.

[Comment.14] Howie: Eventually we will need to state where the
security policy information/DB does get discussed/
specified.

[Comment.15] Editors: Much more text is needed here no doubt.

Authors' Addresses

     Susan Flynn Symington
     The MITRE Corporation
     7515 Colshire Drive
     McLean, VA 22102
     US

     Phone: +1 (703) 983-7209
     Email: susan@mitre.org
     URI: http://mitre.org/

     Stephen Farrell
     Trinity College Dublin
     Distributed Systems Group
     Department of Computer Science
     Trinity College
     Dublin 2
     Ireland

     Phone: +353-1-608-1539
     Email: stephen.farrell@cs.tcd.ie

     Howard Weiss
     SPARTA, Inc.
     7110 Samuel Morse Drive
     Columbia, MD 21046
     US

     Phone: +1-443-430-8089
     Email: hsw@sparta.com

     Peter Lovell
     SPARTA, Inc.
     7110 Samuel Morse Drive
     Columbia, MD 21046
     US

     Phone: +1-443-430-8052
     Email: peter.lovell@sparta.com

Full Copyright Statement

Intellectual Property

Acknowledgment