DTN Research Group
Internet-Draft
Expires: September 4, 2006

S. Symington
The MITRE Corporation
S. Farrell
Trinity College Dublin
H. Weiss
SPARTA, Inc.

March 3, 2006

Bundle Security Protocol Specification
draft-irtf-dtnrg-bundle-security-01

Status of this Memo

This Internet-Draft will expire on September 4, 2006.

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

Editorial Comments . . . .

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 [9].

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 [3] 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 headers used to implement the bundle protocol, excluding the security-specific headers defined here.
>
> The Delay-Tolerant Networking Security Overview [9] provides an informative overview and high-level description of DTN security.

1.2. 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.

[Comment.1]

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-+   +-^---------+
   |Trans1   v |   + ^  T1/T2  v |     + ^  T2/T3  v |   | ^  Trans3 |
   +---------v-+   +-^---------v-+     +-^---------v +   +-^---------+
   |Net1     v |   | ^  N1/N2  v |     | ^  N2/N3  v |   | ^  Net3   |
   +---------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 BPA4 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 header to a bundle

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

Referring to figure 1 again:

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

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

If the bundle that originates at BN1 as source is given a security header 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 header.

A bundle may have multiple security headers. The security-source of a bundle with respect to a given security header in the bundle may be the same as or different from the security-source of the bundle with respect to a different security header in the bundle. Similarly, the security-destination of a bundle with respect to each of that bundle's security headers may be the same or different.

2. Security Headers

There are three types of security headers that MAY be included in a bundle. These are the Bundle Authentication Header (BAH), the Payload Security Header (PSH), and the Confidentiality Header (CH).

> The BAH is used to assure the authenticity of the bundle along a single hop from forwarder to intermediate receiver.
>
> The PSH is used to assure the authenticity of the bundle from the PSH security-source, which creates the PSH, to the PSH security-destination, which verifies the PSH authenticator. The authentication information in the PSH may (if the ciphersuite allows) be verified by any node in between the PSH security-source and the PSH security-destination that has access to the cryptographic keys and revocation status information required to do so.
>
> Since a BAH protects on a "hop-by-hop" basis and a PSH protects on a (sort of) "end-to-end" basis, whenever both are present the BAH MUST form the "outer" layer of protection - that is, the BAH MUST always be calculated and added to the bundle after the PSH has been calculated and added to the bundle.
>
> The CH indicates that some parts of the bundle have been encrypted while en route between the CH security-source and the CH security-destination.

Each of the security headers uses the Canonical Bundle Header Format as defined in the Bundle Protocol Specification. That is, each security header is comprised of the following elements:

- Header type code

- Header processing control flags

- Header data length

- Header-type-specific data fields

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

2.1. Abstract Security Header

An ASH consists of the following mandatory and optional fields:

- Header-type code (one byte) - as in all bundle protocol headers except the primary bundle header. The header types codes for the security headers are:

  BAH: 0x02

  PSH: 0x03

  CH: 0x04

- Header processing control flags (one byte) - as in all bundle protocol headers except the primary bundle header. There are no major constraints on the use of the header processing flags, however, we would expect that the BAH would not have the "transmit status report" bit set since it is only present on a hop-by-hop basis.

  For a PSH, or CH, we would not expect the "replicate in every fragment" bit to be set.[Comment.2]

- Header data length (SDNV) - as in all bundle protocol headers except the primary bundle header. SDNV encoding is described in the bundle protocol.

- Header-type-specific data fields as follows:

  - Ciphersuite ID - identifies the ciphersuite in use. This is two bytes long, though the top five bits are used to indicate the presence or absence of the optional fields below.

  - (optional) Correlator - when more than one related header is inserted then this field must have the same value in each related header instance. This is encoded as an SDNV.

  - (optional) Ciphersuite parameters - parameters to be used with the ciphersuite in use, e.g. a key identifier or initialization vector (IV). This is encoded as an SDNV. The encoding rules for this are defined as part of the ciphersuite specification.

  - (optional) Security-source-length - specifies the length of the following security source field.

- (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. The length of this field in octets is the value of the security-source-length field.

- (optional) Security-destination-length - specifies the length of the following security destination field.

- (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. The length of this field in octets is the value of the security-destination-length field.

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

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

```
+--------+--------+--------+--------+---------+------------+-------+
|type    |flags   |len     | ciphersuite      |correlator  | params|
+--------+--------+--------+--------+---------+------------+-------+
| src-len         | security source                                |
+-----------------+------------------------------------------------+
| dest-len        | security destination                           |
+--------+--------+------------------------------------------------+
| res-len         | security result                                |
+--------+--------+------------------------------------------------+
```

The structure of an abstract security header

Figure 2

The ciphersuite ID is a 16-bit value with the top five bits indicating which of the optional fields are present (value = "1") or absent (value="0"). The remaining 11 bits indicate the ciphersuite.

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.

The structure of the ciphersuite ID bytes 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.

src - the most significant bit (bit 15) indicates whether the ASH contains the optional security-source-length and security-source fields.

dest - bit 14 indicates whether the security-destination-length and security-destination fields are present or not.

parm - bit 13 indicates whether the ASH contains optional ciphersuite parameters or not.

corr - bit 12 indicates whether or not the ASH contains an optional correlator.

res - bit 11 indicates whether or not the ASH contains security result length and security result fields.

bits 10-0 represent the ciphersuite number, giving a maxium of 2048 different ciphersuites.

```
Ciphersuite ID
Bit    Bit   Bit   Bit   Bit   Bit   Bit   Bit
15    14    13    12    11    10     ...          1     0
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|src  |dest |parm |corr |res  | ciphersuite ID              |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Figure 3

A little bit more terminology: when the header is a PSH then we refer to the PSH-source when we mean the security source field in the PSH. Similarly we may refer to the CH-dest, meaning the security-destination field of the CH. For example, referring to Figure 1 again, if the bundle that originates at BN1 as source is given a Confidentiality Header (CH) by BN1 that is protected using a key held by BN3 and it is given a Payload Security Header (PSH) by BN1, then BN1 is both the CH-source and the PSH-source of the bundle, and BN3 is the CH-dest of the bundle.

The correlator field is used to associate several related instances of a security header. This can be used to place a BAH that contains the ciphersuite information at the "front" of a (probably large) bundle , and another correlated BAH 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 BAH. There are

similar use cases for multiple related instances of PSH and CH as will be seen below.

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

2.2. Bundle Authentication Header

In this section we describe typical BAH field values for two scenarios - where a single instance of the BAH 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 BAH is used:

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

The header 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 BAH.

The correlator field MUST NOT be present.

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

The security-source field MUST be present and MUST identify the forwarder of the bundle.

The security-destination field SHOULD NOT be present unless the ciphersuite requires this information (since the first node receiving the bundle ought be the one to validate the BAH).

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 BAH instances, the first instance is as defined above, except the correlator MUST be present and the security result MUST be absent. The second instance of the BAH MUST have the same correlator value present and MUST contain a security

result. The other optional fields MUST NOT be present. Typically, this second instance of a BAH will be the last header of the bundle.

2.3. Payload Security Header

A PSH is an ASH with the following additional restrictions:

The header type code value MUST be 0x03.

The header 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 PSH be present in the bundle. The correlator MUST NOT be present if the ciphersuite only requires one instance of the PSH in the bundle.

The ciphersuite parameters field MAY be present.

The security-source field MAY be present.

The security-destination field 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 BAH, this field MUST be present if the correlator is absent. If more than one related instance of the PSH is required then this is handled in the same way as described for the BAH above.

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.

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

2.4. Confidentiality Header

A typical confidentiality ciphersuite will encrypt the payload using a randomly generated bundle encrypting key (BEK) and will use a CH

security result to carry the BEK encrypted with some long term key encryption key (KEK). or well-known public key. If neither the destination or 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 CH security results will contain headers encrypted using the BEK if non-payload headers are to be encrypted.

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

Payload super-encryption is allowed. If a CH ciphersuite supports such super-encryption, then the ciphersuite MUST provide an unambiguous way to do the decryption operations in the correct order (e.g. by encoding the "layer" information as a ciphersuite parameter). This "in-place" encryption of payload bytes is so as to allow bundle payload fragmentation and re-assembly to operate without knowledge of whether encryption has occurred. A side-effect of this "in-place" encryption is that the payload will typically be expanded by up-to a ciphertext blocksize if the bulk cipher is a block cipher. Another is that the 2nd application of confidentiality does not generally protect the parameters of the first which represent a vulnerability in some circumstances.

A CH is an ASH with the following additional restrictions:

The header type code value MUST be 0x04.

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

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

The correlator MUST be present if more than one related CH instance is required. More than one CH instance might be required if the payload is to be encrypted for more than one security-destination so as to be robust in the face of routing uncertainties. These multiple CH instances, however, would not be related and would therefore not require correlators. On the other hand, multiple related CH instances would be required if both the payload and the PSH headers in the bundle were to be encrypted. These multiple CH instances would require correlators to associate them with each other.

The ciphersuite parameters field MAY be present

The security-source field MAY be present.

The security-destination field MAY be present (and typically will be).

The security result MAY be present and normally represents an encrypted bundle encryption key or encrypted versions of bundle headers other than the payload header.

As was the case for the BAH and PSH, if the ciphersuite requires more than one instance of the CH, then the first occurrence MUST contain most of the optional fields, (e.g. security destination etc.) and subsequent instances MUST NOT contain those same fields. Unlike the case for the BAH and PSH, however, the security result field MAY (and probably will) be included in multiple related CH instances.

Put another way: when a node is encrypting some (non-payload) headers, it MUST first create a CH with the required ciphersuite ID, parameters etc. as specified above. Typically, this CH will appear "early" in the bundle. If this "first" CH doesn't contain all of the ciphertext, then it may be followed by other (correlated) CH, which MUST NOT repeat the ciphersuite or other information from the first CH.

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

Since a single CH security result can contain the ciphertext for multiple (non-payload) plaintext headers, the node simply catenates these plaintext headers prior to encryption. After decryption the recovered plaintext should then replace the CH in the bundle for futher processing (e.g. PSH verification). This recovered plaintext MUST contain all the appropriate header type, processing flags and length information. In other words delete the CH in question and place the recovered plaintext, which consists of additional (non-payload) headers, in the bundle at the location from which the CH was deleted.

Even if a to-be-encrypted header has the "discard" flag set, whether or not the CH's "discard" flag is set is an implementation/policy decision for the encrypting node. (The "discard" flag is more properly called the "discard if header cannot be processed" flag.)[Comment.4]

2.5. PSH and CH combinations

Given the above definitions, nodes are free to combine applications of PSH and CH 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 PSH security result containing a signature in clear, this would allow payload guesses to 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 PSH or CH, must be applied to all instances with the same correlator.

We now list the PSH and CH 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 CH, but with the payload encrypted multiple times, "in-place".
>
> Multiple parallel authenticators - a single security source might wish to integrity protect a bundle in multiple ways, in the hope that one of them will be useful. This could be required if the path the bundle will take 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 PSH which presumably all protect the payload, and which cannot in general protect one another. Note that this logic can also apply to a BAH, if the unpredictable routing happens in the convergence layer, so we also envisage support for multiple parallel uses of BAH.
>
> Multiple sequential authenticators - if some security destination requires assurance about the route that bundles have taken, then it might insist on each forwarding node adding its own PSH. 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 PSH's will likely be layered, so that each protects the

earlier applications of PSH.

Authenticated and encrypted bundles - a single bundle may require both authenticity and confidentiality. In this case, most 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.[Comment.5]

There are no doubt other valid ways to combine PSH and CH instances, but these are the "core" set we wish to support. 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. If a particular transmission request satisfies the node's policy 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.6]

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 PSH. Similarly it is also valid to implement a node which always rejects all requests which imply the use of a PSH.

Nodes MUST consult their security policy to determine the criteria that a received bundle ought to meet before it will be forwarded. These criteria MAY include a determination of whether or not the received bundle must include valid BAH, PSH or CH. 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, destined for the forwarding node's own endpoint.[Comment.7]

The node's policy MAY call for the node to add or subtract some security headers, for example, requiring the node attempt to encrypt (parts of) the bundle for some security-destination, or requiring that the node add a PSH. If the node's policy requires a BAH 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 headers in the bundle.

3.2. 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. 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 BAH or PSH) must be defined.

This section defines two bundle canonicalisation algorithms which can be used by various ciphersuites. [Comment.8]

3.2.1. Strict canonicalisation

The first algorithm which can be used basically permits no changes at all to the bundle between when it is forwarded at the security-source and when it is received at the security-destination and is mainly intended for use in BAH ciphersuites. This algorithm simply involves catenating all headers in the order presented, but omits all security result fields which are present in headers of the ciphersuite type in question - that is, when a BAH ciphersuite specifies this algorithm then we omit all BAH security results, when a PSH ciphersuite specifies this algorithm then we omit all PSH security results.

Notes:

- In the above we call for security results to be omitted, this means that no bytes at all are input - we do not set the security result length to zero or any equivalent.

- The 'res' bit of the ciphersuite ID which indicates that a security value will be present is part of the canonical form.

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

3.2.2. **Mutable canonicalisation**

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

The basic approach is to define a canonical form for the primary header, and catenate that with the security and payload headers in the order that they will be transmitted. This algorithm ignores all other headers on the basis that we cannot tell whether or not they are liable to change as the bundle transits the network.

The canoncial form of the primary header is shown below.

Essentially, it de-references the dictionary header, adjusts lengths where necessary and ignores flags that may change in transit.

```
+----------------+----------------+----------------+----------------+
|    Version     |  Proc. Flags   |   COS Flags    |   SRR Flags    |
+----------------+----------------+---------------------------------+
|                Canonical primary header 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 (8 bytes)                   +
|                                                                   |
+---------------------------------+---------------------------------+
|                             Lifetime                              |
+----------------+----------------+----------------+----------------+
|                      Fragment offset (optional)                   |
+----------------+----------------+---------------------------------+
|            Total application data unit length (optional)          |
+----------------+----------------+---------------------------------+
```

The canonical form of the primary bundle header.

Figure 4

The fields shown are:

Version, Processing Flags, COS, SRR - are all copied from the first four bytes of the primary header and will contain the version and the immutable flag values from the primary header. Formed by copying the processing, COS and SRR flag fields from the

primary header and then subequently setting all of the mutable bits to zero. This requires ANDing with the (four byte) value 0xFF1E031F so that the mutable and reserved bits are set to zero. The only currently mutable bit masked out here is the "bundle is a fragment" bit - all others are reserved bits.

There is an issue here which PSH ciphersuites MUST tackle. If a bundle is fragmented before the PSH is applied then the PSH 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 PSH. For this reason, PSH ciphersuites which support applying a PSH to fragments MUST specify which bytes of the bundle payload are protected as part of the security parameters. When verifying such a fragment only the bytes from the fragment are input to the PSH verification. Of course, if is also valid for a ciphersuite to be specified so as to only apply to entire bundles and not to fragments.

Length - a four-byte value containing the length (in bytes) of this structure.[Comment.9]

Destination endpoint ID length and value - are the length (as a four byte value) and value of the destination endpoint ID from the primary bundle header. The URI is simply copied from the relevant part(s) of the dictionary header and is not itself canonicalised.

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 and Lifetime are simply copied from the primary header.

Fragment offset and Total application data unit length are copied from the primary header if they are present there (which is controlled by one of the flags).

Payload headers are generally copied as-is, with the exception that the processing flags value in the canonical version MUST be ANDed with 0x17 to ensure that currently "reserved" bits are clear, but that the "last header" flag is ignored. The reason to ignore the "last header" flag is that if I create a bundle with both PSH and BAH, then the next hop will remove BAH instances, and if one of those was the last header, it may set the "last header" flag of e.g. a PSH instance. The "last header" is therefore a mutable bit and should be

omitted from the canonical form.

Another exception occurs in cases where only a fragment of the payload was protected, when only those bytes of the payload header payload field are considered part of the canonical form.

Security headers are handled likewise, with the exception that the ciphersuite will likely specify that the "current" security header security result field not be considered part of the canonical form. This differs from the case in strict canonicalisation since we might use the mutable canonicalisation algorithm to handle sequential signatures, where later signatures should cover earlier ones.

Notes:

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

- We omit the reserved flags on the basis that we cannot determine whether or not they will change in transit. This means that this algorithm may have to be revised if those flags are given a definition and if we want to protect them.

- 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.

- Note that the URI encoding above will be a cause for errors if any node rewrites the dictionary for example changing the DNS part of some HTTP URL from lower-case to upper case. Since there are no exact general rules for canonicalising URIs (or even worse IRIs), we simply have to live with this problem.

- All length fields here are four byte values in network byte order. We do not need to optimise the encoding since the values are never sent over the network.

3.3. Source confidentiality

Since every bundle MUST contain a primary header that cannot be encrypted, and which contains the source endpoint ID (and others), if we want to provide source confidentiality, then we have to invent a fake primary header with false values for these fields and then a new header type to contain the actual values.

Similarly, there may be confidentiality requirements applying to other parts of the primary header (e.g. the current-custodian) and we support these in the same way.

Since we don't know if we'll do this...details are TBD:-)[Comment.10]

3.4. 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 BAH. If the bundle is not required to have a BAH, then BAH processing on the received bundle is complete and the bundle is ready to be further processed for CH/PSH handling or delivery or forwarding.

If the bundle is required to have a BAH but it does not, then the bundle MUST be discarded and processed no further; in this case a bundle status report indicating the authentication failure MAY be generated, destined for the receiving node's own endpoint.

Otherwise, if the bundle does have a BAH, then the value in the security result field of the BAH of the received bundle MUST be verified according to the ciphersuite specification. If the check fails for all BAHs in the bundle, the bundle has failed to authenticate and the bundle SHALL be discarded and processed no further; in this case, a bundle status report indicating the authentication failure MAY be generated, destined for the intermediate receiver's own endpoint. Otherwise, if any of the BAHs present verify, the bundle is ready to have its CH processed (if it includes one).

When forwarding a bundle that included some BAHs when it was received, these BAHs MUST be stripped from the bundle. New BAHs MAY be added as required by policy. This might require correcting the "last header" field of the to-be-forwarded bundle.

If the bundle has a CH and the receiving node is the CH destination for the bundle (either because the node is listed in the bundle's CH-dest field or because the node is listed as the bundle's destination and there is no CH-dest field), the node MUST decrypt the relevant parts of the bundle according to the ciphersuite specification and delete the CH in question. If the CH security result included the ciphertext of anything other than an encrypted BEK that was used to encrypt the bundle payload, the recovered plaintext headers MUST be placed in the bundle at the location from which the CH was deleted.

If the bundle has a PSH and the receiving node is the PSH destination for the bundle (either because the node is listed in the bundle's PSH-dest field or because the node is listed as the bundle's destination and there is no PSH-dest field), the node MUST verify the value in the security result field of the PSH according to the ciphersuite specification. If the check fails, the bundle has failed

to authenticate and the bundle SHALL be discarded and processed no further; a bundle status report indicating the failure MAY be generated, destined for the receiving node's own endpoint. Otherwise, if the PSH verifies, the bundle is ready to be processed for either delivery or forwarding. Before forwarding the bundle, the node SHOULD remove the PSH from the bundle, unless there is the likelihood that some downstream node will also be able to verify the PSH.

If the bundle has a PSH and the receiving node is not the PSH-dest for the bundle but the ciphersuite allows, the receiving node MAY, if it is able, verify the value in the security result field. If the check fails, the node SHALL discard the bundle and it MAY send a bundle status report indicating the failure to the receiving node's own endpoint.

3.5. The At-Most-Once-Delivery Option

An application may request (in some 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.

We define a new option for use in such cases, known as "at-most-once-delivery". If this option is chosen, then the application is indicating that the node node SHALL check for replayed bundles, discard duplicates, and deliver at most one copy of each received bundle to the application. If this option is not chosen, the application is indicating that node SHALL deliver all received bundle copies to the application. If this option is not chosen, the node MAY (subject to policy) deliver all bundles or else behave as if the option had been chosen.

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 should uniquely identify a bundle, has been received before. 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.

The duration for which a node maintains entries on such a list is an implementation matter.

If any application has indicated that it wants a node to use the "at most once" delivery option for a particular destination endpoint ID that is in a bundle, then the node MUST compare the (source, timestamp) pair values of the bundle with the local list of such

values of already-received bundles.

3.6. Bundle Fragmentation and Reassembly

If it is necessary for a node to fragment a bundle and security is being used on that bundle, the following security-specific processing is REQUIRED:

Firstly, a BAH, PSH or CH MUST NOT be fragmented. At this time, only the payload field of the payload header MAY be fragmented.

If the bundle is required by the security policy to have a BAH before being forwarded, all fragments resulting from that bundle MUST contain individual BAH values.

If the original bundle had a PSH, then each of the PSH instances MUST be included in some fragment. A single PSH instance MUST NOT be sent more than once.

If the original bundle had a CH, then the each of the CH instances MUST be included in some fragment. A single CH instance MUST NOT be sent more than once.

3.7. Reactive fragmentation

When original bundle transmission is terminated before the entire bundle has been transmitted, the receiving node SHALL consult its security policy to determine whether it is permitted to transform the received portion of the bundle into a bundle fragment for further forwarding. Whether or not such reactive fragmentation is permitted SHALL be dependent on the security policy in combination with the ciphersuite used to calculate the BAH authentication information if required.

In such cases, if the original bundle is fragmented by the intermediate receiver (reactively), and the BAH-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 an original bundle transmission is terminated before the entire bundle has been transmitted, if the truncated bundle arriving at the intermediate receiver is reactively fragmented and forwarded, only the part of the bundle that was not received MUST be retransmitted. Before retransmitting this portion of the bundle, it SHALL be changed into a fragment and, if the original bundle included a BAH, the fragmented bundle MUST also, and its BAH SHALL be recalculated.

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

4. Mandatory Ciphersuites

This section defines the mandatory ciphersuites for this specification. There is currently one mandatory ciphersuite for each of BAH, PSH and CH. The BAH ciphersuite is based on shared secrets using HMAC. The PSH ciphersuite is based on digital signatures using RSA with SHA256. The CH ciphersuite is based on using RSA for key transport and AES for bulk encryption.

4.1. BAH-HMAC

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

Security parameters are optional with this scheme, but if used then the value of the security parameter MUST be used as a key identifier. In the absence of a key identifier the intermediate receiver is expected to be able to find the correct key based on the sending identity (from the security-source and/or convergence layer).

BAH-HMAC uses the strict canonicalisation algorithm in Section 3.2.1.

The variant of HMAC to be used is HMAC-SHA1 as defined in RFC 2104 [4].[Comment.11]

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

This means that normally, the BAH will be the second and last headers of the bundle. If a forwarder wishes to apply more than one correlated BAH 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" BAHs, and their "at back" "partners" (without any security result), before canonicalising.

Inserting more than one correlated BAH 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 BAH instances MUST be included in the bundle before canonicalisation.

4.2. PSH-RSA-SHA256

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

If the bundle being signed has been fragemented before signing, then we have to specify which bytes were signed, in case the signed bundle is subequently fragmented for a second time. So, if the bundle is a fragment, then the security parameters MUST include two SDNV encoded numbers, representing the offset and length of the signed fragment. If the entire bundle is signed then these numbers MUST be omitted.

The security 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 PSH 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 PSH verification.

PSH-RSA-SHA256 uses the mutable canonicalisation algorithm Section 3.2.2. 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 PSH.

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

"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. CH-RSA-AES-PAYLOAD-PSH

[Comment.13]

The CH-RSA-AES-PAYLOAD-PSH ciphersuite has cipersuite ID value 0x003.

This scheme only allows for payload and PSH encryption and involves encrypting every instance of a PSH as well as the payload.

This ciphersuite requires the use of a single CH instance if the bundle does not contain a PSH, and multiple CH instances if the bundle includes one or more PSHs. A first CH is created which

contains the encrypted bundle encryption key (BEK). Thereafter each PSH and the payload bytes are encrypted using a key derived from the BEK.

For the first CH, there MUST be a security parameter which contains a 16 byte IV, optionally followed by a key identifier (whose format is again out of scope here). The security result contains the BEK encrypted using PKCSv1.5 rsaEncryption as specified in RFC 3370 [6].

For each subsequent PSH, the entire header is replaced by a CH that is correlated with the first CH and whose security result is the ciphertext form of the PSH, including the header type, etc.

For the payload, only the bytes of the bundle payload field are affected, being replaced by ciphertext.

Since we are separately encrypting the payload and all of the PSH headers present, we would like to use different keys for each encryption. We do this using a key derivation function (KDF) which takes a different input for each header (or part of the header, as in the case of payload encryption) that is to be encrypted. The input to the KDF is the BEK catenated with the following values:

- The IV from the security parameter. [16 bytes]

- The "header number" (counting the primary header as 1, the next as 2 etc regardless of the header type). [1 octet]

- The header type. [1 octet]

- If the header is a PSH (i.e., if it is not the Bundle Payload Header), then the correlator field value. [SDNV]

Note: Identifying the right KDF here is stil TBD. Plan is to see what happens with the recent NIST KDF paper (which is in conflict with KDF's used in various Internet protocols). Expect change.

The output from the KDF is called the result encryption key (REK) and is used to encrypt the payload or PSH bytes.

The REK uses the AES algorithm in CBC mode as specified by the id-aes-cbc object identifier in RFC 3565 [7] [Comment.14][Comment.15]

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 BAH-HMAC ciphersuite.

- The use of well-known RSA public keys for PSH-RSA-SHA256 and CH-RSA-AES-PAYLOAD-PSH ciphersuites.

Since endpoint IDs are URIs and URIs can be placed in X.509 [8] 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 (PEP) 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 BAHs.

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

Under what conditions received bundles SHALL be required to have valid PSHs and/or CHs.

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

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

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

Under what conditions a CH 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.16]

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 BAH which MUST be verified to contain a valid security result. If the bundle does not have a BAH, then the bundle MUST be discarded and processed no further; a bundle status report indicating the authentication failure MAY be generated, destined for the receiving node's own endpoint.

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

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

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

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

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

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 status report indicating the failure SHOULD be generated, destined for the receiving node's own endpoint.

7. Security Considerations

[Comment.17]

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

If a bundle's PSH-dest is not the same as its destination, then some node other than the destination (the node identified as the PSH-dest) is expected to validate the PSH security result while the bundle is en route. However, if for some reason the PSH is not validated, there is no way for the destination to become aware of this. Typically, a PSH-dest will remove the PSH from the bundle after verifying the PSH and before forwarding it. However, if there is a possibility that the PSH will also be verified at a downstream node, the PSH-dest will leave the PSH in the bundle. Therefore, if a destination receives a bundle with a PSH that has a PSH-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 PSH-source but before being received by its PSH-dest, the payload in the bundle MUST be reassembled before validating the PSH security result in order for the security result to validate correctly. Therefore, if the PSH-dest is not capable of performing payload reassembly, its utility as a PSH-dest will be limited to validating only those bundles that have not been fragmented since being forwarded from the PSH-source. Similarly, if a bundle is fragmented after being forwarded by its PSH-source but before being received by its PSH-dest, all fragments MUST be received at that PSH-dest in order for the bundle payload to be able to be reassembled. If not all fragments are received at the PSH-dest node, the bundle will not be able to be authenticated, and will therefore never be forwarded by this PSH-dest node.

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 header 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., "Bundle Protocol Specification", draft-irtf-dtnrg-bundle-spec-04.txt , December 2005.

[3] Cerf, V., "Delay-Tolerant Network Architecture", draft-irtf-dtnrg-arch-04.txt , December 2005, <draft-irtf-dtnrg-arch-02.txt>.

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

[5] 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.

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

[7] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, July 2003.

[8] 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

[9] Farrell, S., Symington, S., and H. Weiss, "Delay-Tolerant Network Security Overview", draft-irtf-dtnrg-sec-overview-01.txt , March 2006.

Editorial Comments

[Comment.1] Editors: There may be some (valid) question marks over this terminology. We weren't entirely sure ourselves.

[Comment.2] Stephen: I guess there could be some weird corner case where a CH 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 PSH/CH.

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

[Comment.4] Howie: It may be nice to be able to signal both confidentiality and authentication using only one header instead of using two separate headers (PSH and CH).

[Comment.5] Add approppriate forward references to ciphersuites defined later.

[Comment.6] 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? I dunno.

[Comment.7] 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.8] Stephen: This is a fairly big new section and so probably contains errors!

[Comment.9] Editors: Check that mask value at the very last moment (incl. during auth-48) to be sure its (still) correct.

[Comment.10] Stephen: Should we support source confidentiality? Might complicate PSH which is the downside IMO.

[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 entire section is to be treated as a strawman for the present.

[Comment.14] Speculation: There would be an interesting possibility opened up were we to use a stream cipher with the REK. That is that we could then encrypt and decrypt independently - Alice could encrypt for Bob, then Charlie could encrypt for Dessie, then Bob could decrypt, then Dessie. Better in terms of not adding padding but worse in that it trivially allows known plaintext manipulation if there's no PSH.

[Comment.15] Editors: Another option might also be to switch to using counter mode rather than CBC which would have the benefit of allowing some fragments to be decrypted even if not all fragments arrive. While that seems nice enough to do, it would of course require us to think more about fragments and so is for the next version if at all.

[Comment.16] Howie: Eventually we will need to state where the security policy information/DB does get discussed/ specified.

[Comment.17] 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.
7075 Samuel Morse Drive
Columbia, MD 21046
US

Phone: +1-410-872-1515 x201
Email: hsw@sparta.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.