Internet Engineering Task Force (IETF)
Request for Comments: 6618
Category: Experimental
ISSN: 2070-1721

J. Korhonen, Ed.
Nokia Siemens Networks
B. Patil
Nokia
H. Tschofenig
Nokia Siemens Networks
D. Kroeselberg
Siemens

May 2012

# Mobile IPv6 Security Framework Using Transport Layer Security for Communication between the Mobile Node and Home Agent

## Abstract

Mobile IPv6 signaling between a Mobile Node (MN) and its Home Agent (HA) is secured using IPsec.  The security association (SA) between an MN and the HA is established using Internet Key Exchange Protocol (IKE) version 1 or 2.  The security model specified for Mobile IPv6, which relies on IKE/IPsec, requires interaction between the Mobile IPv6 protocol component and the IKE/IPsec module of the IP stack. This document proposes an alternate security framework for Mobile IPv6 and Dual-Stack Mobile IPv6, which relies on Transport Layer Security for establishing keying material and other bootstrapping parameters required to protect Mobile IPv6 signaling and data traffic between the MN and HA.

## Status of This Memo

## Copyright Notice

## Table of Contents

## 1. Introduction

Mobile IPv6 (MIPv6) [RFC6275] signaling, and optionally user traffic, between a Mobile Node (MN) and Home Agent (HA) are secured by IPsec [RFC4301]. The current Mobile IPv6 security architecture is specified in [RFC3776] and [RFC4877]. This security model requires a tight coupling between the Mobile IPv6 protocol part and the IKE(v2)/IPsec part of the IP stack. Client implementation experience has shown that the use of IKE(v2)/IPsec with Mobile IPv6 is fairly complex.

This document proposes an alternate security framework for Mobile IPv6 and Dual-Stack Mobile IPv6. The objective is to simplify implementations as well as make it easy to deploy the protocol without compromising on security. Transport Layer Security (TLS) [RFC5246] is widely implemented in almost all major operating systems and extensively used by various applications. Instead of using IKEv2 to establish security associations, the security framework proposed in this document is based on TLS-protected messages to exchange keys and bootstrapping parameters between the MN and a new functional entity called the "Home Agent Controller" (HAC). The Mobile IPv6 signaling between the mobile node and home agent is subsequently secured using the resulting keys and negotiated ciphersuite. The HAC can be co-located with the HA, or it can be an independent entity. For the latter case, communication between the HAC and HA is not defined by this document. Such communication could be built on top of AAA protocols such as Diameter.

The primary advantage of using TLS for the establishment of Mobile IPv6 security associations as compared to the use of IKEv2 is the ease of implementation (especially on the mobile nodes) while providing an equivalent level of security. A solution which decouples Mobile IPv6 security from IPsec, for securing signaling messages and user plane traffic, is proposed herein that reduces client implementation complexity.

The security framework proposed in this document is not intended to replace the currently specified architecture that relies on IPsec and IKEv2. It provides an alternative solution that is more optimal for certain deployment scenarios. This and other alternative security models have been considered by the MEXT working group at the IETF, and it has been decided that the alternative solutions should be published as Experimental RFCs, so that more implementation and deployment experience with these models can be gathered. The status of this proposal may be reconsidered in the future if it becomes clear that it is superior to others.

## 2. Terminology and Abbreviations

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 [RFC2119].

Home Agent Controller (HAC):

The home agent controller is a node responsible for bootstrapping Mobile IPv6 security associations between a mobile node and one or more home agents. The home agent controller also provides key distribution to both mobile nodes and home agents. Mobile IPv6 bootstrapping is also performed by the HA in addition to the security association bootstrapping between the mobile node and home agent controller.

Binding Management Messages:

Mobile IPv6 signaling messages between a mobile node and a home agent, correspondent node, or mobility access point to manage the bindings are referred to as binding management messages. Binding Updates (BUs) and Binding Acknowledgement (BA) messages are examples of binding management messages.

## 3. Background

Mobile IPv6 design and specification began in the mid-to-late 90s. The security architecture of Mobile IPv6 was based on the understanding that IPsec is an inherent and integral part of the IPv6 stack and any protocol that needs security should use IPsec unless there is a good reason not to. As a result of this mindset, the Mobile IP6 protocol relied on the use of IPsec for security between the MN and HA. Reusing security components that are an integral part of the IP stack is a good design objective for any protocol; however, in the case of Mobile IPv6, it increases implementation complexity. It should be noted that Mobile IPv4 [RFC5944], for example, does not use IPsec for security and instead has specified its own security solution. Mobile IPv4 has been implemented and deployed on a reasonably large scale and the security model has proven itself to be sound.

Mobile IPv6 standardization was completed in 2005 along with the security architecture using IKE/IPsec specified in RFC 3776 [RFC3776]. With the evolution to IKEv2 [RFC5996], Mobile IPv6 security has also been updated to rely on the use of IKEv2 [RFC4877]. Implementation exercises of Mobile IPv6 and Dual-Stack Mobile IPv6 [RFC5555] have identified the complexity of using IPsec and IKEv2 in conjunction with Mobile IPv6. Implementing Mobile IPv6 with IPsec and IKEv2 requires modifications to both the IPsec and IKEv2 components, due to the communication models that Mobile IPv6 uses and the changing IP addresses. For further details, see Section 7.1 in [RFC3776].

This document proposes a security framework based on TLS-protected establishment of Mobile IPv6 security associations, which reduces implementation complexity while maintaining an equivalent (to IKEv2/IPsec) level of security.

## 4. TLS-Based Security Establishment

### 4.1. Overview

The security architecture proposed in this document relies on a secure TLS session established between the MN and the HAC for mutual authentication and MN-HA security association bootstrapping. Authentication of the HAC is done via standard TLS operation wherein the HAC uses a TLS server certificate as its credentials. MN authentication is done by the HAC via signaling messages that are secured by the TLS connection. Any Extensible Authentication Protocol (EAP) method or Pre-Shared Key (PSK) can be used for authenticating the MN to the HAC. Upon successful completion of authentication, the HAC generates keys that are delivered to the MN through the secure TLS channel. The same keys are also provided to the assigned HA. The HAC also provides the MN with MIPv6 bootstrapping information such as the IPv6 and IPv4 address of the HA, the home network prefix, the IPv6 and/or IPv4 Home Address (HoA), and DNS server address.

The MN and HA use security associations based on the keys and Security Parameter Indexes (SPIs) generated by the HAC and delivered to the MN and HA to secure signaling and optionally user plane traffic. Figure 1 below is an illustration of the process.

Signaling messages and user plane traffic between the MN and HA are always UDP encapsulated. The packet formats for the signaling and user plane traffic is described in Sections 6.3 and 6.4.

```
   MN                            HAC                 HA
   --                            ---                 --
    |                             |                   |
    | /-------------------------\ |                   |
    |/                           \|                   |
    |\    TLS session setup      /|                   |
    | \-------------------------/ |                   |
    |                             |                   |
    | /-------------------------\ |                   |
    |/     MN Authentication     \|                   |
    |\                           /|                   |
    | \-------------------------/ |                   |
    |                             |                   |
    | /-------------------------\ |                   |
    |/   HAC provisions the MN   \|                   |
    |\  keys, SPI, & MIPv6 parms /|                   |
    | \-------------------------/ |                   |
    |                             |--MNID, keys, SPI->|
    |                             |                   |
    | /--------------------------------------------\  |
    |/     MN-HA SA established; Secures            \ |
    |\     signaling and optionally user traffic    / |
    | \--------------------------------------------/  |
    |                                                 |
    |------------BU(encrypted)----------------------->|
    |                                                 |
    |<---------BAck(encrypted)------------------------|
```

Figure 1: High-Level Architecture

## 4.2. Architecture

The TLS-based security architecture is shown in Figure 2. The signaling message exchange between the MN and the HAC is protected by TLS. It should be noted that an HAC, a AAA server, and an HA are logically separate entities and can be collocated in all possible combinations. There MUST be a strong trust relationship between the HA and the HAC, and the communication between them MUST be both integrity and confidentially protected.

```
   +------+             +------+            +------+
   |Mobile|     TLS     |Home  |    AAA     | AAA  |
   | Node |<----------->|Agent |<---------->|Server|
   |      |             |Contrl|            |      |
   +------+             +------+            +------+
      ^                     ^                   ^
      |                     |                   |
      | BU/BA/../           | e.g., AAA         | AAA
      | (Data)              |                   |
      |                     v                   |
      |                +---------+              |
      |                | MIPv6   |              |
      +--------------->| Home    |<-------------+
                       | Agent(s)|
                       +---------+
```

Figure 2: TLS-Based Security Architecture Overview

## 4.3. Security Association Management

Once the MN has contacted the HAC and mutual authentication has taken place between the MN and the HAC, the HAC securely provisions the MN with all security-related information inside the TLS protected tunnel. This security-related information constitutes a security association (SA) between the MN and the HA. The created SA MUST NOT be tied to the Care-of Address (CoA) of the MN.

The HAC may proactively distribute the SA information to HAs, or the HA may query the SA information from the HAC once the MN contacts the HA. If the HA requests SA information from the HAC, then the HA MUST be able to query/index the SA information from the HAC based on the SPI identifying the correct security association between the MN and the HA.

The HA may want the MN to re-establish the SA even if the existing SA is still valid. The HA can indicate this to the MN using a dedicated Status Code in a BA (value set to REINIT_SA_WITH_HAC). As a result, the MN SHOULD contact the HAC prior to the SA timing out, and the HAC would provision the MN and HAs with a new SA to be used subsequently.

The SA established between MN and HAC SHALL contain at least the following information:

Mobility SPI:

This parameter is an SPI used by the MN and the HA to index the SA between the MN and the HA. The HAC is responsible for assigning SPIs to MNs. There is only one SPI for both binding management messaging and possible user data protection. The same SPI is used for both directions between the MN and the HA. The SPI values are assigned by the HAC. The HAC MUST ensure uniqueness of the SPI values across all MNs controlled by the HAC.

MN-HA keys for ciphering:

A pair of symmetric keys (MN -> HA, HA -> MN) used for ciphering Mobile IPv6 traffic between the MN and the HA. The HAC is responsible for generating these keys. The key generation algorithm is specific to the HAC implementation.

MN-HA shared key for integrity protection:

A pair of symmetric keys (MN -> HA, HA -> MN) used for integrity protecting Mobile IPv6 traffic between the MN and the HA. This includes both binding management messages and reverse-tunneled user data traffic between the MN and the HA. The HAC is responsible for generating these keys. The key generation algorithm is specific to the HAC implementation. In the case of combined algorithms, a separate integrity protection key is not needed and may be omitted, i.e., the encryption keys SHALL be used.

Security association validity time:

This parameter represents the validity time for the security association. The HAC is responsible for defining the lifetime value based on its policies. The lifetime may be in the order of hours or weeks. The MN MUST re-contact the HAC before the SA validity time ends.

Security association scope:

This parameter defines whether the security association is applied to Mobile IPv6 signaling messages only or to both Mobile IPv6 signaling messages and data traffic.

Selected ciphersuite:

This parameter is the ciphersuite used to protect the traffic between the MN and the HA. This includes both binding management messages and reverse-tunneled user data traffic between the MN and the HA. The selected algorithms SHOULD be one of the mutually supported ciphersuites of the negotiated TLS version between the MN and the HAC. The HAC is responsible for choosing the mutually supported ciphersuite that complies with the policy of the HAC. Obviously, the HAs under HAC’s management must have at least one ciphersuite with the HAC in common and need to be aware of the implemented ciphersuites. The selected ciphersuite is the same for both directions (MN -> HA and HA -> MN).

Sequence numbers:

A monotonically increasing unsigned sequence number used in all protected packets exchanged between the MN and the HA in the same direction. Sequence numbers are maintained per direction, so each SA includes two independent sequence numbers (MN -> HA, HA -> MN). The initial sequence number for each direction MUST always be set to 0 (zero). Sequence numbers cycle to 0 (zero) when increasing beyond their maximum defined value.

### 4.4. Bootstrapping of Additional Mobile IPv6 Parameters

When the MN contacts the HAC to distribute the security-related information, the HAC may also provision the MN with various MIPv6-related bootstrapping information. Bootstrapping of the following information SHOULD at least be possible:

Home Agent IP Address:

The IPv6 and IPv4 address of the home agent assigned by the HAC.

Mobile IPv6 Service Port Number:

The port number where the HA is listening to UDP [RFC0768] packets.

Home Address:

The IPv6 and/or IPv4 home address assigned to the mobile node by the HAC.

Home Link Prefix:

Concerns the IPv6 Home link prefix and the associated prefix length.

DNS Server Address:

The address of a DNS server that can be reached via the HA. DNS queries in certain cases cannot be routed to the DNS servers assigned by the access network to which the MN is attached; hence, an additional DNS server address that is reachable via the HA needs to be configured.

The MIPv6-related bootstrapping information is delivered from the HAC to the MN over the same TLS protected tunnel as the security related information.

### 4.5. Protecting Traffic between Mobile Node and Home Agent

The same integrity and confidentiality algorithms MUST be used to protect both binding management messages and reverse-tunneled user data traffic between the MN and the HA. Generally, all binding management messages (BUs, BAs, and so on) MUST be integrity protected and SHOULD be confidentially protected. The reverse-tunneled user data traffic SHOULD be equivalently protected. Generally, the requirements stated in [RFC6275] concerning the protection of the traffic between the MN and the HA also apply to the mechanisms defined by this specification.

## 5. MN-to-HAC Communication

### 5.1. Request-Response Message Framing over TLS-Tunnel

The MN and the HAC communicate with each other using a simple lockstep request-response protocol that is run inside the protected TLS-tunnel. A generic message container framing for the request messages and for the response messages is defined. The message containers are only meant to be exchanged on top of a connection-oriented TLS-layer. Therefore, the end of message exchange is determined by the other end closing the transport connection (assuming the "application layer" has also indicated the completion of the message exchange). The peer initiating the TLS connection is always sending "Requests", and the peer accepting the TLS connection is always sending "Responses". The format of the message container is shown in Figure 3.

All data inside the Content portion of the message container MUST be encoded using octets. Fragmentation of message containers is not supported, which means one request or response at the "application layer" MUST NOT exceed the maximum size allowed by the message container format.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Ver |  Rsrvd  | Identifier    | Length                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Content portion..                                             ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Request-Response Message Container

The 3-bit Ver field identifies the protocol version. The current version is 0, i.e., all bits are set to a value of 0 (zero).

The Rsrvd field MUST be set to a value of 0 (zero),

The Identifier field is meant to match requests and responses. The valid Identifier values are between 1-255. The value 0 MUST NOT be used. The first request for each communication session between the MN and the HAC MUST have the Identifier values set to 1.

The Length field tells the length of the Content portion of the container (i.e., Reserved octet, Identifier octet, and Length field are excluded). The Content portion length MUST always be at least one octet and up to 65535 octets. The value is in network order.

### 5.2. Request-Response Message Content Encoding

The encoding of the message content is similar to HTTP header encoding and complies with the augmented Backus-Naur Form (BNF) defined in Section 2.1 of [RFC2616]. All presented hexadecimal numbers are in network byte order. From now on, we use the TypeValue header (TV-header) term to refer to request-response message content HTTP-like headers.

### 5.3. Request-Response Message Exchange

The message exchange between the MN and the HAC is a simple lockstep request-response type as stated in Section 5.1. A request message includes a monotonically increasing Identifier value that is copied to the corresponding response message. Each request MUST have a different Identifier value. Hence, a reliable connection-oriented transport below the message container framing is assumed. The number of request-response message exchanges MUST NOT exceed 255.

Each new communication session between the MN and the HAC MUST reset the Identifier value to 1. The MN is also the peer that always sends only request messages and the HAC only sends response messages. Once the request-response message exchange completes, the HAC and the MN MUST close the transport connection and the corresponding TLS-tunnel.

In the case of an HAC-side error, the HAC MUST send a response back to an MN with an appropriate status code and then close the transport connection.

The first request message - MHAuth-Init - (i.e., the Identifier is 1) MUST always contain at least the following parameters:

MN-Identity - See Section 5.5.1.

Authentication Method - See Section 5.5.2.

The first response message - MHAuth-Init - (i.e., the Identifier is 1) MUST contain at minimum the following parameters:

Selected authentication Method - See Section 5.5.2.

The last request message from the MN side - MHAuth-Done - MUST contain the following parameters:

Security association scope - See Section 5.6.4.

Proposed ciphersuites - See Section 5.6.5.

Message Authenticator - See Section 5.5.5.

The last response message - MHAuth-Done - that ends the request-response message exchange MUST contain the following parameters:

Status Code - See Section 5.5.4.

Message Authenticator - See Section 5.5.5.

In the case of successful authentication, the following additional parameters:

Selected ciphersuite - See Section 5.6.5.

Security association scope - See Section 5.6.4.

The rest of the security association data - See Section 5.6.

## 5.4. Home Agent Controller Discovery

All bootstrapping information, whether for setting up the SA or for bootstrapping MIPv6-specific information, is exchanged between the MN and the HAC using the framing protocol defined in Section 5.1. The IP address of the HAC MAY be statically configured in the MN or alternatively MAY be dynamically discovered using DNS. In the case of DNS-based HAC discovery, the MN queries either an A/AAAA or a SRV record for the HAC IP address. The actual domain name used in queries is up to the deployment to decide and out of scope of this specification.

## 5.5. Generic Request-Response Parameters

The grammar used in the following sections is the augmented Backus-Naur Form (BNF), the same as that used by HTTP [RFC2616].

### 5.5.1. Mobile Node Identifier

An identifier that identifies an MN. The Mobile Node Identifier is in the form of a Network Access Identifier (NAI) [RFC4282].

```
mn-id = "mn-id" ":" RFC4282-NAI CRLF
```

### 5.5.2. Authentication Method

The HAC is the peer that mandates the authentication method. The MN sends its authentication method proposal to the HAC. The HAC, upon receipt of the MN proposal, returns the selected authentication method. The MN MUST propose at least one authentication method. The HAC MUST select exactly one authentication method or return an error and then close the connection.

```
auth-method = "auth-method" ":" a-method *("," a-method) CRLF
a-method =
     "psk" ; PSK-based authentication
   | "eap" ; EAP-based authentication
```

### 5.5.3. Extensible Authentication Protocol Payload

Each Extensible Authentication Protocol (EAP) [RFC3748] message is an encoded string of hexadecimal numbers. The "eap-payload" is completely transparent as to which EAP-method or EAP message is carried inside it. The "eap-payload" can appear in both request and response messages:

```
eap-payload = "eap-payload" ":" 1*(HEX HEX) CRLF
```

### 5.5.4. Status Code

The "status-code" MUST only be present in the response message that ends the request-response message exchange. The "status-code" follows the principles of HTTP and the definitions found in Section 10 of RFC 2616 also apply for these status codes listed below:

```
status-code = "status-code" ":" status-value CRLF
status-value =
     "100" ; Continue
   | "200" ; OK
   | "400" ; Bad Request
   | "401" ; Unauthorized
   | "500" ; Internal Server Error
   | "501" ; Not Implemented
   | "503" ; Service Unavailable
   | "504" ; Gateway Time-out
```

### 5.5.5. Message Authenticator

The "auth" header contains data used for authentication purposes. It MUST be the last TV-header in the message and calculated over the whole message till the start of the "msg-header":

```
msg-auth = "auth" ":" 1*(HEX HEX) CRLF
```

### 5.5.6. Retry After

```
retry-after = "retry-after" ":" rfc1123-date CRLF
```

### 5.5.7. End of Message Content

```
end-of-message = 2CRLF
```

### 5.5.8. Random Values

Random numbers generated by the MN and the HAC, respectively. The length of the random number MUST be 32 octets (before TV-header encoding):

```
mn-rand = "mn-rand" ":" 32(HEX HEX) CRLF
hac-rand = "hac-rand" ":" 32(HEX HEX) CRLF
```

## 5.6. Security Association Configuration Parameters

During the Mobile IPv6 bootstrapping, the MN and the HAC negotiate a single ciphersuite for protecting the traffic between the MN and the HA. The allowed ciphersuites for this specification are a subset of those in TLS version 1.2 (see Appendix A.5 of [RFC5246]) per Section 5.6.5. This might appear as a constraint as the HA and the HAC may have implemented different ciphersuites. These two nodes are, however, assumed to belong to the same administrative domain. In order to avoid exchanging supported MN-HA ciphersuites in the MN-HAC protocol and to reuse the TLS ciphersuite negotiation procedure, we make this simplifying assumption. The selected ciphersuite MUST provide integrity and confidentiality protection.

Section 5.6.5 provides the mapping from the TLS ciphersuites to the integrity and encryption algorithms allowed for MN-HA protection. This mapping mainly ignores the authentication algorithm part that is not required within the context of this specification. For example, [RFC5246] defines a number of AES-based ciphersuites for TLS including ’TLS_RSA_WITH_AES_128_CBC_SHA’. For this specification, the relevant part is ’AES_128_CBC_SHA’.

All the parameters described in the following sections apply only to a request-response protocol response message to the MN. The MN has no way of affecting the provisioning decision of the HAC.

### 5.6.1. Security Parameter Index

The 28-bit unsigned SPI number identifies the SA used between the MN and the HA. The value 0 (zero) is reserved and MUST NOT be used. Therefore, values ranging from 1 to 268435455 are valid.

The TV-header corresponding to the SPI number is as follows:

```
mip6-spi = "mip6-spi" ":" 1*DIGIT CRLF
```

### 5.6.2. MN-HA Shared Keys

The MN-HA shared integrity (ikey) and encryption (ekey) keys are used to protect the traffic between the MN and the HA. The length of these keys depend on the selected ciphersuite.

The TV-headers that carry these two parameters are the following:

```
mip6-mn-to-ha-ikey = "mip6-mn-to-ha-ikey" ":" 1*(HEX HEX) CRLF
mip6-ha-to-mn-ikey = "mip6-ha-to-mn-ikey" ":" 1*(HEX HEX) CRLF
mip6-mn-to-ha-ekey = "mip6-mn-to-ha-ekey" ":" 1*(HEX HEX) CRLF
mip6-ha-to-mn-ekey = "mip6-ha-to-mn-ekey" ":" 1*(HEX HEX) CRLF
```

### 5.6.3. Security Association Validity Time

The end of the SA validity time is encoded using the "rfc1123-date" format, as defined in Section 3.3.1 of [RFC2616].

The TV-header corresponding to the SA validity time value is as follows:

```
mip6-sa-validity-end = "mip6-sa-validity-end" ":" rfc1123-date CRLF
```

### 5.6.4. Security Association Scope (SAS)

The SA is applied either to Mobile IPv6 signaling messages only or to both Mobile IPv6 signaling messages and data traffic. This policy MUST be agreed between the MN and HA prior to using the SA. Otherwise, the receiving side will be unaware of whether the SA applies to data traffic and hence unable to decide how to act when receiving unprotected packets of PType 1 (see Section 6.4).

```
mip6-sas = "mip6-sas" ":" 1DIGIT CRLF
```

where a value of "0" indicates that the SA does not protect data traffic and a value of "1" indicates that all data traffic MUST be protected by the SA. If the mip6-sas value of an SA is set to 1, any packet received with a PType value that does not match the mip6-sas value of the SA MUST be silently discarded.

The HAC is the peer that mandates the used security association scope. The MN sends its proposal to the HAC, but eventually the security association scope returned from the HAC defines the used scope.

### 5.6.5. Ciphersuites and Ciphersuite-to-Algorithm Mapping

The ciphersuite negotiation between HAC and MN uses a subset of the TLS 1.2 ciphersuites and follows the TLS 1.2 numeric representation defined in Appendix A.5 of [RFC5246]. The TV-headers corresponding to the selected ciphersuite and ciphersuite list are the following:

```
mip6-ciphersuite = "mip6-ciphersuite" ":" csuite CRLF
csuite = "{" suite "}"
suite =
     "00" "," "02" ; CipherSuite NULL_SHA           = {0x00,0x02}
   | "00" "," "3B" ; CipherSuite NULL_SHA256        = {0x00,0x3B}
   | "00" "," "0A" ; CipherSuite 3DES_EDE_CBC_SHA   = {0x00,0x0A}
   | "00" "," "2F" ; CipherSuite AES_128_CBC_SHA    = {0x00,0x2F}
   | "00" "," "3C" ; CipherSuite AES_128_CBC_SHA256 = {0x00,0x3C}

mip6-suitelist = "mip6-suitelist" ":" csuite *("," csuite) CRLF
```

All other Ciphersuite values are reserved.

The following integrity algorithms MUST be supported by all implementations:

| | |
|---|---|
| HMAC-SHA1-96 | [RFC2404] |
| AES-XCBC-MAC-96 | [RFC3566] |

The binding management messages between the MN and HA MUST be integrity protected. Implementations MUST NOT use a NULL integrity algorithm.

The following encryption algorithms MUST be supported:

| | |
|---|---|
| NULL | [RFC2410] |
| TripleDES-CBC | [RFC2451] |
| AES-CBC with 128-bit keys | [RFC3602] |

Traffic between MN and HA MAY be encrypted. Any integrity-only Ciphersuite makes use of the NULL encryption algorithm.

Note: This document does not consider combined algorithms. The following table provides the mapping of each ciphersuite to a combination of integrity and encryption algorithms that are part of the negotiated SA between MN and HA.

| Ciphersuite | Integ. Algorithm | Encr. Algorithm |
|---|---|---|
| NULL_SHA | HMAC-SHA1-96 | NULL |
| NULL_SHA256 | AES-XCBC-MAC-96 | NULL |
| 3DES_EDE_CBC_SHA | HMAC-SHA1-96 | TripleDES-CBC |
| AES_128_CBC_SHA | HMAC-SHA1-96 | AES-CBC with 128-bit keys |
| AES_128_CBC_SHA256 | AES-XCBC-MAC-96 | AES-CBC with 128-bit keys |

Ciphersuite-to-Algorithm Mapping

### 5.7. Mobile IPv6 Bootstrapping Parameters

In parallel with the SA bootstrapping, the HAC SHOULD provision the MN with relevant MIPv6-related bootstrapping information.

The following generic BNFs are used to form IP addresses and prefixes. They are used in subsequent sections.

```
ip6-addr   = 7( word ":" ) word CRLF
word       = 1*4HEX
ip6-prefix = ip6-addr "/" 1*2DIGIT
ip4-addr   = 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT
ip4-subnet = ip4-addr "/" 1*2DIGIT
```

#### 5.7.1. Home Agent Address

The HAC MAY provision the MN with an IPv4 or an IPv6 address of an HA, or both.

```
mip6-haa-ip6 = "mip6-haa-ip6" ":" ip6-addr CRLF
mip6-haa-ip4 = "mip6-haa-ip4" ":" ip4-addr CRLF
```

#### 5.7.2. Mobile IPv6 Service Port Number

The HAC SHOULD provision the MN with an UDP port number, where the HA expects to receive UDP packets. If this parameter is not present, then the IANA reserved port number (mipv6tls) MUST be used instead.

```
mip6-port = "mip6-port" ":" 1*5DIGIT CRLF
```

#### 5.7.3. Home Addresses and Home Network Prefix

The HAC MAY provision the MN with an IPv4 or an IPv6 home address, or both. The HAC MAY also provision the MN with its home network prefix.

```
      mip6-ip6-hoa = "mip6-ip6-hoa" ":" ip6-addr CRLF
      mip6-ip4-hoa = "mip6-ip4-hoa" ":" ip4-addr CRLF
      mip6-ip6-hnp = "mip6-ip6-hnp" ":" ip6-prefix CRLF
      mip6-ip4-hnp = "mip6-ip4-hnp" ":" ip4-subnet CRLF
```

#### 5.7.4. DNS Server

The HAC may also provide the MN with DNS server configuration options. These DNS servers are reachable via the home agent.

```
      dns-ip6 = "dns-ip6" ":" ip6-addr CRLF
      dns-ip4 = "dns-ip4" ":" ip4-addr CRLF
```

### 5.8. Authentication of the Mobile Node

This section describes the basic operation required for the MN-HAC mutual authentication and the channel binding. The authentication protocol described as part of this section is a simple exchange that follows the Generalized Pre-Shared Key (GPSK) exchange used by EAP-GPSK [RFC5433]. It is secured by the TLS tunnel and is cryptographically bound to the TLS tunnel through channel binding based on [RFC5056] and on the channel binding type ’tls-server-endpoint’ described in [RFC5929]. As a result of the channel binding type, this method can only be used with TLS ciphersuites that use server certificates and the Certificate handshake message. For example, TLS ciphersuites based on PSK or anonymous authentication cannot be used.

The authentication exchange MUST be performed through the encrypted TLS tunnel. It performs mutual authentication between the MN and the HAC based on a PSK or based on an EAP-method (see Section 5.9). Note that an HAC MUST NOT allow MNs to renegotiate TLS sessions. The PSK protocol is described in this section. It consists of the message exchanges (MHAuth-Init, MHAuth-Mid, MHAuth-Done) in which both sides exchange nonces and their identities, and compute and exchange a message authenticator ’auth’ over the previously exchanged values, keyed with the pre-shared key. The MHAuth-Done messages are used to deal with error situations. Key binding with the TLS tunnel is ensured by channel binding of the type "tls-server-endpoint" as described by [RFC5929] where the hash of the TLS server certificate serves as input to the ’auth’ calculation of the MHAuth messages.

Note: The authentication exchange is based on the GPSK exchange used by EAP-GPSK. In comparison to GPSK, it does not support exchanging an encrypted container (it always runs through an already protected TLS tunnel). Furthermore, the initial request of the authentication exchange (MHAuth-Init) is sent by the MN (client side) and is comparable to EAP-Response/Identity, which reverses the roles of request and response messages compared to EAP-GPSK. Figure 4 shows a successful protocol exchange.

```
MN                                                       HAC
 |                                                         |
 | Request/MHAuth-Init (...)                               |
 |-------------------------------------------------------->|
 |                                                         |
 |                           Response/MHAuth-Init (...) |
 |<--------------------------------------------------------|
 |                                                         |
 | Request/MHAuth-Done (...)                               |
 |-------------------------------------------------------->|
 |                                                         |
 |                           Response/MHAuth-Done (...) |
 |<--------------------------------------------------------|
 |                                                         |
```

Figure 4: Authentication of the Mobile Node Using Shared Secrets

1) Request/MHAuth-Init: (MN -> HAC)

   mn-id, mn-rand, auth-method=psk

2) Response/MHAuth-Init: (MN <- HAC)

   [mn-rand, hac-rand, auth-method=psk, [status],] auth

3) Request/MHAuth-Done: (MN -> HAC)

   mn-rand, hac-rand, sa-scope, ciphersuite-list, auth

4) Response/MHAuth-Done: (MN <- HAC)

   [sa-scope, sa-data, ciphersuite, bootstrapping-data,] mn-rand, hac-rand, status, auth

Where ’auth’ for MN -> HAC direction is as follows:

```
auth = HMAC-SHA256(PSK, "MN" | msg-octets | CB-octets)
```

Where ’auth’ for MN <- HAC direction is as follows:

```
auth = HMAC-SHA256(PSK, "HAC" | msg-octets | CB-octets)
```

In the above, "MN" is 2 ASCII characters without null termination and "HAC" is 3 ASCII characters without null termination.

The length "mn-rand", "hac-rand" is 32 octets. Note that "|" indicates concatenation and optional parameters are shown in square brackets [..]. The square brackets can be nested.

The shared secret PSK can be variable length. ’msg-octets’ includes all payload parameters of the respective message to be signed except the ’auth’ payload. CB-octets is the channel binding input to the auth calculation that is the "TLS-server-endpoint" channel binding type. The content and algorithm (only required for the "TLS-server-endpoint" type) are the same as described in [RFC5929].

The MN starts by selecting a random number ’mn-rand’ and choosing a list of supported authentication methods coded in ’auth-method’. The MN sends its identity ’mn-id’, ’mn-rand’, and ’auth-method’ to the HAC in MHAuth-Init. The decision of which authentication method to offer and which to pick is policy and implementation dependent and, therefore, outside the scope of this document.

In MHAuth-Done, the HAC sends a random number ’hac-rand’ and the selected ciphersuite. The selection MUST be one of the MN-supported ciphersuites as received in ’ciphersuite-list’. Furthermore, it repeats the received parameters of the MHAuth-Init message ’mn-rand’. It computes a message authenticator ’auth’ over all the transmitted parameters except ’auth’ itself. The HAC calculates ’auth’ over all parameters and appends it to the message.

The MN verifies the received Message Authentication Code (MAC) and the consistency of the identities, nonces, and ciphersuite parameters transmitted in MHAuth-Auth. In case of successful verification, the MN computes a MAC over the session parameter and returns it to the HAC in MHAuth-Done. The HAC verifies the received MAC and the consistency of the identities, nonces, and ciphersuite parameters transmitted in MHAuth-Init. If the verification is successful, MHAuth-Done is prepared and sent by the HAC to confirm successful completion of the exchange.

## 5.9. Extensible Authentication Protocol Methods

Basic operation required for the MN-HAC mutual authentication using EAP-based methods.

```
   MN                                                      HAC
    |                                                       |
    | Request/MHAuth-Init (...)                             |
    |------------------------------------------------------>|
    |                                                       |
    |                            Response/MHAuth-Init (..., |
    |                     eap-payload=EAP-Request/Identity) |
    |<------------------------------------------------------|
    |                                                       |
    | Request/MHAuth-Mid (eap-payload=                      |
    |              EAP-Response/Identity)                   |
    |------------------------------------------------------>|
    |                                                       |
    |     Response/MHAuth-Mid (eap-payload=EAP-Request/...) |
    |<------------------------------------------------------|
    |                                                       |
    :                                                       :
    :        ..EAP-method specific exchanges..              :
    :                                                       :
    |                                                       |
    | Request/MHAuth-Done (eap-payload=EAP-Response/...,    |
    |                     ..., auth)                        |
    |------------------------------------------------------>|
    |                                                       |
    |        Response/MHAuth-Done (eap-payload=EAP-Success, |
    |                              ..., auth)               |
    |<------------------------------------------------------|
    |                                                       |
```

Figure 5: Authentication of the Mobile Node Using EAP

1) Request/MHAuth-Init: (MN -> HAC)

   mn-id, mn-rand, auth-method=eap

2) Response/MHAuth-Init: (MN <- HAC)

   [auth-method=eap, eap, [status,]] auth

3) Request/MHAuth-Mid: (MN -> HAC)

   eap, auth

4) Response/MHAuth-Mid: (MN <- HAC)

eap, auth

MHAuth-Mid exchange is repeated as many times as needed by the used EAP-method.

5) Request/MHAuth-Done: (MN -> HAC)

sa-scope, ciphersuite-list, eap, auth

6) Response/MHAuth-Done: (MN <- HAC)

[sa-scope, sa-data, ciphersuite, bootstrapping-data,] eap, status, auth

Where ’auth’ for MN -> HAC direction is as follows:

auth = HMAC-SHA256(shared-key, "MN" | msg-octets | CB-octets)

Where ’auth’ for MN <- HAC direction is as follows:

auth = HMAC-SHA256(shared-key, "HAC" | msg-octets | CB-octets)

In the above, "MN" is 2 ASCII characters without null termination and "HAC" is 3 ASCII characters without null termination.

In MHAuth-Init and MHAuth-Mid messages, shared-key is set to "1". If the EAP-method is key-deriving and creates a shared Master Session Key (MSK) as a side effect of Authentication shared-key MUST be the MSK in all MHAuth-Done messages. This MSK MUST NOT be used for any other purpose. In case the EAP method does not generate an MSK, shared-key is set to "1".

’msg-octets’ includes all payload parameters of the respective message to be signed except the ’auth’ payload. CB-octets is the channel binding input to the AUTH calculation that is the "TLS-server-endpoint" channel binding type. The content and algorithm (only required for the "TLS-server-endpoint" type) are the same as described in [RFC5929].

# 6. Mobile Node to Home Agent Communication

## 6.1. General

The following subsections describe the packet formats used for the traffic between the MN and the HA. This traffic includes binding management messages (for example, BU and BA messages), reverse-tunneled and encrypted user data, and reverse-tunneled plaintext user data. This specification defines a generic packet format, where everything is encapsulated inside UDP. See Sections 6.3 and 6.4 for detailed illustrations of the corresponding packet formats.

The Mobile IPv6 service port number is where the HA expects to receive UDP packets. The same port number is used for both binding management messages and user data packets. The reason for multiplexing data and control messages over the same port number is due to the possibility of Network Address and Port Translators located along the path between the MN and the HA. The Mobile IPv6 service MAY use any ephemeral port number as the UDP source port, and it MUST use the Mobile IPv6 service port number as the UDP destination port. The Mobile IPv6 service port is dynamically assigned to the MN during the bootstrapping phase (i.e., the mip6-port parameter) or, in absence of the bootstrapping parameter, the IANA reserved port (mipv6tls) MUST be used.

The encapsulating UDP header is immediately followed by a 4-bit Packet Type (PType) field that defines whether the packet contains an encrypted mobility management message, an encrypted user data packet, or a plaintext user data packet.

The Packet Type field is followed by a 28-bit SPI value, which identifies the correct SA concerning the encrypted packet. For any packet that is neither integrity protected nor encrypted (i.e., no SA is applied by the originator), the SPI MUST be set to 0 (zero). Mobility management messages MUST always be at least integrity protected. Hence, mobility management messages MUST NOT be sent with an SPI value of 0 (zero).

There is always only one SPI per MN-HA mobility session and the same SPI is used for all types of protected packets independent of the direction.

The SPI value is followed by a 32-bit Sequence Number value that is used to identify retransmissions of protected messages (integrity protected or both integrity protected and encrypted, see Figures 7 and 8) . Each endpoint in the security association maintains two "current" Sequence Numbers: the next one to be used for a packet it initiates and the next one it expects to see in a packet from the other end. If the MN and the HA ends initiate very different numbers of messages, the Sequence Numbers in the two directions can be very different. In the case data protection is not used (see Figure 9), the Sequence Number MUST be set to 0 (zero). Note that the HA SHOULD initiate a re-establishment of the SA before any of the Sequence Number cycle.

Finally, the Sequence Number field is followed by the data portion, whose content is identified by the Packet Type. The data portion may be protected.

## 6.2. PType and Security Parameter Index

The PType is a 4-bit field that indicates the Packet Type (PType) of the UDP encapsulated packet. The PType is followed by a 28-bit SPI value. The PType and the SPI fields are treated as one 32-bit field during the integrity protection calculation.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PType |                        SPI                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Security Parameter Index with Packet Type

A SPI value of 0 (zero) indicates a plaintext packet. If the packet is integrity protected or both integrity protected and encrypted, the SPI value MUST be different from 0. When the SPI value is set to 0, then the PType MUST also be 0.

## 6.3. Binding Management Message Formats

The binding management messages that are only meant to be exchanged between the MN and the HA MUST be integrity protected and MAY be encrypted. They MUST use the packet format shown in Figure 7.

All packets that are specific to the Mobile IPv6 protocol, contain a Mobility Header (as defined in Section 6.1.1. of RFC 6275) and are used between the MN and the HA shall use the packet format shown in Figure 7. (This means that some Mobile IPv6 mobility management messages, such as the Home Test Init (HoTI) message, are treated as data packets and using encapsulation described in Section 6.4 and shown in Figures 8 and 9).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:         IPv4 or IPv6 header (src-addr=Xa, dst-addr=Ya)        :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:            UDP header (src-port=Xp,dst-port=Yp)               :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|PType=8|                    SPI                                | ^Int.
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Cov-
|                      Sequence Number                          | |ered
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | ----
|                    Payload Data  (variable)                   | |   ^
:                                                               : |   |
|                                                               | |Conf.
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Cov-
|               |     Padding (0-255 bytes)                     | |ered
+-+-+-+-+-+-+-+-+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |   |
|                               |  Pad Length   | Next Header   | v   v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|         Integrity Check Value-ICV   (variable)                |
:                                                               :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: UDP-Encapsulated Binding Management Message Format

The PType value 8 (eight) identifies that the UDP-encapsulated packet contains a Mobility Header (defined by RFC 6275) and other relevant IPv6 extension headers. Note, there is no additional IP header inside the encapsulated part. The Next Header field MUST be set to value of the first encapsulated header. The encapsulated headers follow the natural IPv6 and Mobile IPv6 extension header alignment and formatting rules.

The Padding, Pad Length, Next Header, and ICV fields follow the rules of Section 2.4 to 2.8 of [RFC4303] unless otherwise stated in this document. For an SPI value of 0 (zero) that indicates an unprotected packet, the Padding, Pad Length, Next Header, and ICV fields MUST NOT be present.

The source and destination IP addresses of the outer IP header (i.e., the src-addr and the dst-addr in Figure 7) use the current CoA of the MN and the HA address.

### 6.4. Reverse-Tunneled User Data Packet Formats

There are two types of reverse-tunneled user data packets between the MN and the HA: those that are integrity protected and/or encrypted and those that are sent in the clear. The MN or the HA decides whether to apply integrity protection and/or encryption to a packet or to send it in the clear based on the mip6-sas value in the SA. If the mip6-sas is set to 1, the originator MUST NOT send any user data packets in the clear, and the receiver MUST silently discard any packet with the PType set to 0 (unprotected). It is RECOMMENDED that confidentiality and integrity protection of user data traffic be applied. The reverse-tunneled IPv4 or IPv6 user data packets are encapsulated as is inside the ’Payload Data’ shown in Figures 8 and 9.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:         IPv4 or IPv6 header (src-addr=Xa, dst-addr=Ya)        :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:            UDP header (src-port=Xp,dst-port=Yp)               :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|PType=1|                    SPI                                | ^Int.
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Cov-
|                      Sequence Number                          | |ered
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | ----
|                     Payload Data  (variable)                  | |   ^
:                                                               : |   |
|                                                               | |Conf.
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Cov-
|               |     Padding (0-255 bytes)                     | |ered
+-+-+-+-+-+-+-+-+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |   |
|                               |  Pad Length   | Next Header   | v   v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|         Integrity Check Value-ICV   (variable)                |
:                                                               :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: UDP-Encapsulated Protected User Data Packet Format

The PType value 1 (one) identifies that the UDP-encapsulated packet contains an encrypted-tunneled IPv4/IPv6 user data packet. The Next Header field header MUST be set to value corresponding the tunneled IP packet (e.g., 41 for IPv6).

The Padding, Pad Length, Next Header, and ICV fields follow the rules of Section 2.4 to 2.8 of [RFC4303] unless otherwise stated in this document. For an SPI value of 0 (zero) that indicates an unprotected packet, the Padding, Pad Length, Next Header, and ICV fields MUST NOT be present.

The source and destination IP addresses of the outer IP header (i.e., the src-addr and the dst-addr in Figure 8) use the current CoA of the MN and the HA address. The ESP-protected inner IP header, which is not shown in Figure 8, uses the home address of the MN and the correspondent node (CN) address.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   :         IPv4 or IPv6 header (src-addr=Xa, dst-addr=Ya)        :
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   :            UDP header (src-port=Xp,dst-port=Yp)               :
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |PType=0|                        0                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                0                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   :           Payload Data (plain IPv4 or IPv6 Packet)            :
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: UDP-Encapsulated Non-Protected User Data Packet Format

The PType value 0 (zero) identifies that the UDP-encapsulated packet contains a plaintext-tunneled IPv4/IPv6 user data packet. Also, the SPI and the Sequence Number fields MUST be set to 0 (zero).

The source and destination IP addresses of the outer IP header (i.e., the src-addr and the dst-addr in Figure 9) use the current CoA of the MN and the HA address. The plaintext inner IP header uses the home address of the MN and the CN address.

# 7. Route Optimization

Mobile IPv6 route optimization as described in [RFC6275] is not affected by this specification. Route optimization is possible only between an IPv6 MN and CN. UDP encapsulation of signaling and data traffic is only between the MN and HA. The return routability signaling messages such as HoTI/HoT and CoTI/CoT [RFC6275] are treated as data packets and encapsulation, when needed, is per the description in Section 6.4 of this specification. The data packets between an MN and CN that have successfully completed the return routability test and created the appropriate entries in their binding cache are not UDP encapsulated using the packet formats defined in this specification but follow the [RFC6275] specification.

# 8. IANA Considerations

## 8.1. New Registry: Packet Type

IANA has created a new registry under the [RFC6275] Mobile IPv6 parameters registry for the Packet Type as described in Section 6.1.

| Description | Value |
|---|---|
| non-encrypted IP packet | 0 |
| encrypted IP packet | 1 |
| mobility header | 8 |

Following the allocation policies from [RFC5226], new values for the Packet Type AVP MUST be assigned based on the "RFC Required" policy.

## 8.2. Status Codes

A new Status Code (to be used in BA messages) is reserved for the cases where the HA wants to indicate to the MN that it needs to re-establish the SA information with the HAC. The following value is reserved in the [RFC6275] Status Codes registry:

| | |
|---|---|
| REINIT_SA_WITH_HAC | 176 |

## 8.3. Port Numbers

A new port number (mipv6tls) for UDP packets is reserved from the existing PORT NUMBERS registry.

mipv6tls 7872

## 9. Security Considerations

This document describes and uses a number of building blocks that introduce security mechanisms and need to interwork in a secure manner.

The following building blocks are considered from a security point of view:

1. Discovery of the HAC

2. Authentication and MN-HA SA establishment executed between the MN and the HAC (PSK- or EAP-based) through a TLS tunnel

3. Protection of MN-HA communication

4. AAA interworking

### 9.1. Discovery of the HAC

No dynamic procedure for discovering the HAC by the MN is described in this document. As such, no specific security considerations apply to the scope of this document.

### 9.2. Authentication and Key Exchange Executed between the MN and the HAC

This document describes a simple authentication and MN-HA SA negotiation exchange over TLS. The TLS procedures remain unchanged; however, channel binding is provided.

Authentication: Server-side certificate-based authentication MUST be performed using TLS version 1.2 [RFC5246]. The MN MUST verify the HAC’s TLS server certificate, using either the subjectAltName extension [RFC5280] dNSName identities as described in [RFC6125] or subjectAltName iPAddress identities. In case of iPAddress identities, the MN MUST check the IP address of the TLS connection against these iPAddress identities and SHOULD reject the connection if none of the iPAddress identities match the connection. In case of dNSName identities, the rules and guidelines defined in [RFC6125] apply here, with the following considerations:

- Support for DNS-ID identifier type (the dNSName identity in the subjectAltName extension) is REQUIRED in the HAC and the MN TLS implementations.

- DNS names in the HAC server certificates MUST NOT contain the wildcard character "*".

- The CN-ID MUST NOT be used for authentication within the rules described in [RFC6125].

- The MN MUST set its "reference identifier" to the DNS name of the HAC.

The client-side authentication may depend on the specific deployment and is therefore not mandated. Note that TLS-PSK [RFC4279] cannot be used in conjunction with the methods described in Sections 5.8 and 5.9 of this document due to the limitations of the channel binding type used.

Through the protected TLS tunnel, an additional authentication exchange is performed that provides client-side or mutual authentication and exchanges SA parameters and optional configuration data to be used in the subsequent protection of MN-HA communication. The additional authentication exchange can be either PSK-based (Section 5.8) or EAP-based (Section 5.9). Both exchanges are always performed within the protected TLS tunnel and MUST NOT be used as standalone protocols.

The simple PSK-based authentication exchange provides mutual authentication and follows the GPSK exchange used by EAP-GPSK [RFC5433] and has similar properties, although some features of GPSK like the exchange of a protected container are not supported.

The EAP-based authentication exchange simply defines message containers to allow carrying the EAP packets between the MN and the HAC. In principle, any EAP method can be used. However, it is strongly recommended to use only EAP methods that provide mutual authentication and that derive keys including an MSK in compliance with [RFC3748].

Both exchanges use channel binding with the TLS tunnel. The channel binding type 'TLS-server-endpoint' per [RFC5929] MUST be used.

Dictionary Attacks: All messages of the authentication exchanges specified in this document are protected by TLS. However, any implementation SHOULD assume that the properties of the authentication exchange are the same as for GPSK [RFC5433], in case the PSK-based method per Section 5.8 is used, and are the same as those of the underlying EAP method, in case the EAP-based exchange per Section 5.9 is used.

Replay Protection: The underlying TLS protection provides protection against replays.

Key Derivation and Key Strength: For TLS, the TLS-specific considerations apply unchanged. For the authentication exchanges defined in this document, no key derivation step is performed as the MN-HA keys are generated by the HAC and are distributed to the MN through the secure TLS connection.

Key Control: No joint key control for MN-HA keys is provided by this version of the specification.

Lifetime: The TLS-protected authentication exchange between the MN and the HAC is only to bootstrap keys and other parameters for usage with MN-HA security. The SAs that contain the keys have an associated lifetime. The usage of Transport Layer Security (TLS) Session Resumption without Server-Side State, described in [RFC5077], provides the ability for the MN to minimize the latency of future exchanges towards the HA without having to keep state at the HA itself.

Denial-of-Service (DoS) Resistance: The level of resistance against DoS attacks SHOULD be considered the same as for common TLS operation, as TLS is used unchanged. For the PSK-based authentication exchange, no additional factors are known. For the EAP-based authentication exchange, any considerations regarding DoS resistance specific to the chosen EAP method are expected to be applicable and need to be taken into account.

Session Independence: Each individual TLS protocol run is independent from any previous exchange based on the security properties of the TLS handshake protocol. However, several PSK- or EAP-based authentication exchanges can be performed across the same TLS connection.

Fragmentation: TLS runs on top of TCP and no fragmentation-specific considerations apply to the MN-HAC authentication exchanges.

Channel Binding: Both the PSK and the EAP-based exchanges use channel binding with the TLS tunnel. The channel binding type ’TLS-server-endpoint’ per [RFC5929] MUST be used.

Fast Reconnect: This protocol provides session resumption as part of TLS and optionally the support for [RFC5077]. No fast reconnect is supported for the PSK-based authentication exchange. For the EAP-based authentication exchange, availability of fast reconnect depends on the EAP method used.

Identity Protection: Based on the security properties of the TLS tunnel, passive user identity protection is provided. An attacker acting as man-in-the-middle in the TLS connection would be able to observe the MN identity value sent in MHAuth-Init messages.

Protected Ciphersuite Negotiation: This protocol provides ciphersuite negotiation based on TLS.

Confidentiality: Confidentiality protection of payloads exchanged between the MN and the HAC are protected with the TLS Record Layer. TLS ciphersuites with confidentiality and integrity protection MUST be negotiated and used in order to exchange security sensitive material inside the TLS connection.

Cryptographic Binding: No cryptographic bindings are provided by this protocol specified in this document.

Perfect Forward Secrecy: Perfect forward secrecy is provided with appropriate TLS ciphersuites.

Key confirmation: Key confirmation of the keys established with TLS is provided by the TLS Record Layer when the keys are used to protect the subsequent TLS exchange.

## 9.3. Protection of MN and HA Communication

Authentication: Data origin authentication is provided for the communication between the MN and the HA. The chosen level of security of this authentication depends on the selected ciphersuite. Entity authentication is offered by the MN to HAC protocol exchange.

Dictionary Attacks: The concept of dictionary attacks is not applicable to the MN-HA communication as the keying material used for this communication is randomly created by the HAC and its length depends on the chosen cryptographic algorithms.

Replay Protection: Replay protection for the communication between the MN and the HA is provided based on sequence numbers and follows the design of IPsec ESP.

Key Derivation and Key Strength: The strength of the keying material established for the communication between the MN and the HA is selected based on the negotiated ciphersuite (based on the MN-HAC exchange) and the key created by the HAC. The randomness requirements for security described in [RFC4086] are applicable to the key generation by the HAC.

Key Control: The keying material established during the MN-HAC protocol exchange for subsequent protection of the MN-HA communication is created by the HA and therefore no joint key control is provided for it.

Key Naming: For the MN-HA communication, the security associations are indexed with the help of the SPI and additionally based on the direction (inbound communication or outbound communication).

Lifetime: The lifetime of the MN-HA security associations is based on the value in the mip6-sa-validity-end header field exchanged during the MN-HAC exchange. The HAC controls the SA lifetime.

DoS Resistance: For the communication between the MN and the HA, there are no heavy cryptographic operations (such as public key computations). As such, there are no DoS concerns.

Session Independence: Sessions are independent from each other when new keys are created via the MN-HAC protocol. A new MN-HAC protocol run produces fresh and unique keying material for protection of the MN-HA communication.

Fragmentation: There is no additional fragmentation support provided beyond what is offered by the network layer.

Channel Binding: Channel binding is not applicable to the MN-HA communication.

Fast Reconnect: The concept of fast reconnect is not applicable to the MN-HA communication.

Identity Protection: User identities SHOULD NOT be exchanged between the MN and the HA. In the case where binding management messages contain the user identity, the messages SHOULD be confidentiality protected.

Protected Ciphersuite Negotiation: The MN-HAC protocol provides protected ciphersuite negotiation through a secure TLS connection.

Confidentiality: Confidentiality protection of payloads exchanged between the MN and the HAC (for Mobile IPv6 signaling and optionally for the data traffic) is provided utilizing algorithms negotiated during the MN-HAC exchange.

Cryptographic Binding: No cryptographic bindings are provided by this protocol specified in this document.

Perfect Forward Secrecy: Perfect forward secrecy is provided when the MN bootstraps new keying material with the help of the MN-HAC protocol (assuming that a proper TLS ciphersuite is used).

Key Confirmation: Key confirmation of the MN-HA keying material conveyed from the HAC to the MN is provided when the first packets are exchanged between the MN and the HA (in both directions as two different keys are used).

### 9.4. AAA Interworking

The AAA backend infrastructure interworking is not defined in this document and is therefore out of scope.

## 10. Acknowledgements

The authors would like to thank Pasi Eronen, Domagoj Premec, Julien Laganier, Jari Arkko, Stephen Farrell, Peter Saint-Andre and Christian Bauer for their comments.

## 11. References

### 11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.

[RFC2410] Glenn, R. and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, November 1998.

[RFC2451] Pereira, R. and R. Adams, "The ESP CBC-Mode Cipher Algorithms", RFC 2451, November 1998.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3566] Frankel, S. and H. Herbert, "The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec", RFC 3566, September 2003.

[RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, September 2003.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, November 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", RFC 5929, July 2010.

[RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

### 11.2. Informative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC3776] Arkko, J., Devarapalli, V., and F. Dupont, "Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents", RFC 3776, June 2004.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC4877] Devarapalli, V. and F. Dupont, "Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture", RFC 4877, April 2007.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

[RFC5433] Clancy, T. and H. Tschofenig, "Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK) Method", RFC 5433, February 2009.

[RFC5555] Soliman, H., "Mobile IPv6 Support for Dual Stack Hosts and Routers", RFC 5555, June 2009.

[RFC5944] Perkins, C., "IP Mobility Support for IPv4, Revised", RFC 5944, November 2010.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

## Authors' Addresses

Jouni Korhonen (editor)
Nokia Siemens Networks
Linnoitustie 6
Espoo  FIN-02600
Finland

EMail: jouni.nospam@gmail.com

Basavaraj Patil
Nokia
6021 Connection Drive
Irving, TX  75039
USA

EMail: basavaraj.patil@nokia.com

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo  02600
Finland

Phone: +358 (50) 4871445
EMail: Hannes.Tschofenig@gmx.net

Dirk Kroeselberg
Siemens
Otto-Hahn-Ring 6
Munich  81739
Germany

EMail: dirk.kroeselberg@siemens.com