Mobility Extensions (MEXT)	B. Patil
Internet-Draft	Nokia
Intended status: Experimental Protocol	C. Perkins
Expires: April 13, 2012	Tellabs
	H. Tschofenig
	Nokia Siemens Networks
	D. Premec
	Unaffiliated
	October 11, 2011

Problems with the use of IPsec as the security protocol for Mobile IPv6
draft-patil-mext-mip6issueswithipsec-04.txt

Abstract

Mobile IPv6 as specified in RFC3775 relies on IPsec for securing the signaling messages and user plane traffic between the mobile node and home agent. An IPsec SA between the mobile node and the home agent provides security for the mobility signaling. Use of IPsec for securing the data traffic between the mobile node and home agent is optional. This document analyses the implications of the design decision to mandate IPsec as the default security protocol for Mobile IPv6 and consequently Dual-stack Mobile IPv6 and recommends revisiting this decision in view of the experience gained from implementation and adoption in other standards bodies.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Mobile IPv6 as specified in [RFC3775] requires an IPsec security association between the mobile node (MN) and home agent (HA). The IPsec SA protects the mobility signaling messages between the MN and HA. The user data may be optionally protected by the IPsec SA but is not required. The use of IPsec by most hosts today is primarily as a solution for enterprise connectivity through VPN applications. IPsec has not evolved into a generic security protocol.
The use of IPsec and IKE (v1 and v2) with Mobile IPv6 are specified in RFCs 3776 [RFC3776] and 4877 [RFC4877]. The Mobile IP and MIP6 working groups in the IETF chose to mandate IPsec as the default security protocol for Mobile IPv6 based on various criteria and lengthy discussions that occured between the years 2000 and 2004.
Implementation experience with Mobile IPv6 and the security variants with which it has been specified in some SDOs indicates a need to revisit the design choice for MIP6 signaling security. The analysis and recommendation to revisit the security protocol architecture for MIP6 should not be interpreted as a recommendation for Authentication Protocol for Mobile IPv6 [RFC4285]. The objective is to highlight the misfit of IPsec and IKEv2 as the security protocol for MIP6 and hence the need for considering alternatives. A simpler security architecture for securing the signaling and traffic between the MN and HA can co-exist with the IPsec based solution as well.
The objective of Mobile IPv6 [RFC3775] is to enable IP mobility for IPv6 hosts. The security aspect of the protocol is a critical component for consideration in terms of deployment and operation on large scales. If complexity of implementation were a consideration then the current specification dealing with Mobile IPv6, i.e RFC3775 [RFC3775] and RFC5555 [RFC5555] would win high accolades. An implementer spends 20% of his time on implementing the Mobile IPv6 protocol and 80% of the time integrating it with IPsec and IKEv2. And even after that interoperability of the client with home agents is not guaranteed. The IPsec/IKEv2 security architecture may work in implementations wherein the OS, the IPsec/IKEv2 stack and mobile ipv6 client software are all implemented by a single entity. It just does not work on open systems.

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].
This document refers to [RFC3775][RFC4877] for terminology.

3. Background

IP mobility support in IPv6 was considered to be an integral feature of the IPv6 stack based on the experience gained from developing mobility support for IPv4. The design of Mobile IPv6 was worked on by the Mobile IP WG in the late 90s and by the MIP6 WG until its publication as [RFC3775] in 2004.
IPsec [RFC4301] was also intended to be a default component of the IPv6 stack and was the preferred protocol choice for use by any other IPv6 protocol that needed security. Rather than design security into every protocol feature, the intent was to reuse a well-defined security protocol to meet the security needs. Hence Mobile IPv6 has been designed with a security architecture that relies on reusing IPsec.
The Mobile IPv6 specification [RFC3775] was published along with the companion specification "Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents", [RFC3776]. The establishment of the IPsec SA between the MN and HA as per RFC 3776 is based on the use of IKE. The use of IKE in the context of Mobile IPv6 for IPsec SA establishment did not gain traction because of factors such as complexity of IKE and the IETF transitioning to IKEv2. The MIP6 WG completed the specification, Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture [RFC4877] in April 2007. This [RFC4877] is considered as the default security protocol solution for MIP6 and updates [RFC3776].

4. Problem statement

4.1. Problem Statement

Mobile IPv6 is encumbered by its reliance on IPsec [RFC4301] from an implementation and deployment perspective. As a protocol solution for host based mobility, MIP6 can be simpler without the IPsec baggage. The issues with IPsec are even more exacerbated in the case of dual-stack MIP6 [RFC5555].
IPsec SAs between the MN and HA are established either manually or via the use of IKEv2 [RFC4306]. Manual SA configuration is not a scalable solution and hence MIP6 hosts and Home agents rely on IKEv2 for establishing dynamically IPsec SAs. As a result MIP6 depends on the existence of IPsec and IKEv2 for successful operation.
IPsec is unable to provide security protection for MIP6 in a transparent way, and numerous interactions between the host's security subsystems and the MIP6 application are needed in the course of the

regular operation of the MIP6 application. Besides requiring an extensive communications channel between the security subsystems and the MIP6 application, those interactions often also require modification of the MNs security subsystems code. The situation today is such that the communications channel between the IPsec subsystems and the MIP6 application is non existent and this is generally true for most of the commercially available platforms. Even if such a channel were to be available, there does not exist a standardized protocol over that channel which would enable the MIP6 application to communicate with the security modules in a non- implementation specific way. Considering a third party application developer who would like to provide a MIP6 application for a particular platform, the need for numerous interactions with the IPsec subsystem and the unavailability of the standardized communications channel through which such interactions could take place is a major obstacle to the implementation of the mobility protocol. Without such a communication channel being available it is not possible to implement a MIP6 application as a third party developer.

Even if the platform would provide such a communication interface for the MIP6 daemon, this is still insufficient as the MIP6 protocol standardized today [RFC3775] requires numerous changes to the host's IPsec and IKEv2 implementation. This document enumerates various implementation issues related to the interactions between the MIP6 application and the host's security subsystems.

An argument can be made that the MIP6 application itself should provide the required changes to the IPsec subsystems of the platform (maybe in the form of patches). While this is possible at least for some open source platforms to provide modifications to the host's IPsec implementation as well as the key management application(s), this is still an issue for several reasons:

- Target platform could be a commercial platform for which no source code for the security modules (IPsec and IKEv2) is available.

- If the MIP6 application were to patch the IPsec subsystems, then multiple MIP6 applications from different developers would implement it in different ways, which would inevitably lead to variations and problems with interoperability at a minimum, for instance when the user tries to install several MIP6 applications it is difficult to determine which one is the best suited for his/ her needs.

- Key management daemons are usually provided as third party software for which no source code may be available, even if the platform itself is available as open source.

- Even if the MIP6 application developer would be willing to provide patches for the key management daemon to make it work

with his MIP6 application, how would the MIP6 application developer know which of the several available key management daemons the user is running?

*Each application would be able to work only with a single key management daemon, namely the one for which the MIP6 application provides the patches. The user may be running another key management daemon and may be unwilling to change its daemon to the one that comes as part of the MIP6 application.

*Patches for the IPsec part in the kernel and the key management daemon would typically be valid only for the particular version of the kernel and the key management daemon for which they were written. This might prevent the user from upgrading the platform or applying OS security patches that are provided as part of the regular platform maintenance since this would in all probability make the MIP6 application defunct.

*Modifying the security subsystems by a third party is a security issue and users are generally advised to refrain from allowing the security subsystems to be modified in any way.

*he developer may not have the knowledge or the time to modify the platform's IKEv2 and IPsec subsystems, although it might be perfectly capable to deliver the MIP6 application itself.

*There could be copyright issues as well that would prevent modifications of the platform's security subsystems or the delivery of the modifications by the third party.

*Even if the MIP6 application developer is able to come up with the necessary patches for the security subsystem, it is not realistic to expect the prospective user of MIPv6 to first patch the kernel and the key management daemons before using the MIPv6 service.

The above discussion shows why it is unrealistic to expect that the MIP6 application could provide the needed modifications to the IKEv2 and IPsec subsystems of the host. Section 6 presents a more technical discussion of various implementation issues related to the interworking between the MIP6 application and the IPsec/key management modules. The problem in a nutshell for MIP6 is the dependence on IPsec and IKEv2 for successful operation.

4.2. General issues with the use of IPsec for MIP6 security

This section captures several issues with the use of IPsec by MIP6.

1. The design of Mobile IPv6 emphasized the reuse of IPv6 features such as IPsec. IPsec for IPv4 was a bolt-on solution. With the

increasing need for security, IPv6 designers chose to incorporate IPsec as a default feature. There existed an assumption in the MIP6 working group that every IPv6 host would have IPsec capability as a standard feature. While this is true in many host implementations today, the existence of IPsec in every IPv6 stack is not a given. Hence a host which needs to implement Mobile IPv6 must ensure that IPsec and IKEv2 are also available. As a result of this dependence, MIP6 is no longer a standalone host-based mobility protocol. A good example of a host based mobility protocol that works as a self-sufficient module is Mobile IPv4 [RFC3344]. The security associated with MIP4 signaling is integrated into the protocol itself. MIP4 has been successfully deployed on a large scale in several networks.

2. IPsec use in most hosts is generally for the purpose of VPN connectivity to enterprises. It has not evolved into a generic security protocol that can be used by Mobile IPv6 easily. While [RFC4877] does specify the details which enable only the MIP6 signaling to be encapsulated with IPsec, the general method of IPsec usage has been such that all traffic between a host and the IPsec gateway is carried via the tunnel. Selective application of IPsec to protocols is not the norm. Use of IPsec with Mobile IPv6 requires configuration which in many cases is not easily achievable because of reasons such as enterprise environments preventing changes to IPsec policies.

3. A MIP6 home agent is one end of the IPsec SA in a many-to-one relationship. A MIP6 HA may support a very large number of mobile nodes which could be in the hundreds of thousands to millions. The ability to terminate a large number of IPsec SAs (millions) requires signifiant hardware and platform capability. The cost issues of such an HA are detrimental and hence act as a barrier to deployment.

4. The implementation complexity of Mobile IPv6 is greatly increased because of the interaction with IKEv2. The complexity of the protocol implementation is even more so in the case of Dual stack MIP6 [RFC5555] wherein NAT traversal scenarios are considered.

5. IPsec and IKEv2 are not implemented or available by default in every IPv6 or dual stack host. Mobile IPv6 support on such devices is not an option. Many low-end cellular hosts have IP stacks. The need for IPsec and IKEv2 in these devices is not important whereas mobility support is needed in many cases. A simpler security protocol than the use of IPsec/IKEv2 would make MIP6 much more attractive to implement and deploy.

6. [RFC4877] which specifies the use of IKEv2 and IPsec with Mobile IPv6 essentially results in a variant of IPsec which is specific to Mobile IP. Hence this results in added complexity to implementations.

7. Mobile IPv6 needs to be capable of being deployed in situations where alternative security mechanisms are already well-understood by the network administration. It should be possible to enable Mobile IPv6 to work in situations where alternative security mechanisms already supply the necessary authentication and privacy.

8. IPsec has been successfully applied to VPN and other infrastructure operations, but not for general end-to-end applications. Thus, the granularity for selectors was originally not at all sufficient for Mobile IPv6.

9. The way that the IPsec code sits in the usual kernel, and the access mechanisms for the SA database, are not very convenient for use by straightforward implementations of Mobile IPv6. Unusual calling sequences and parameter passing seems to be required on many platforms.

10. In certain environments the use of IPsec and IKEv2 for establishing the SA is considered as an overhead. Bandwidth constrained links such as cellular networks and air interfaces which are in the licensed spectrum tend to be optimized for user traffic. MIP6 signaling with the IPsec overhead and the IKEv2 messages are viewed negatively. It is more acceptable to have signaling without IPsec encapsulation.

The issues listed above can be speculatively attributed as some of the causes for MIP6 not being implemented widely.

4.3. Security Association Management

Once the MN has contacted the HAC and mutual authentication has taken place between the MN and the HAC inside the TLS protected tunnel, the HAC 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 under its management, or the HA may query the SA information from the HAC once the MN contacts the HA. If the HA queries for the SA information from the HAC, then the HA MUST be able to query/index the SA information from the HAC based on the Security Parameter Index (SPI).
In certain situations, 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 would contact the HAC prior the SA times out, and the HAC would provision the MN and HAs with a new SA information.
The SA contains 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 shared key for ciphering:

This parameter is a key used for ciphering Mobile IPv6 traffic between the MN and the HA. The HAC is responsible for generating this key. The key generation algorithm is specific to the HAC implementation.

MN-HA shared key for integrity protection:

This parameter is a key 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 this key. The key generation algorithm is specific to the HAC implementation. In case of combined algorithms a separate integrity protection key is not needed and may be omitted.

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.

Sequence number:

This parameter represents a monotonically increasing unsigned sequence number used in all protected packets exchanged between the MN and the HA. The initial sequence number MUST always be set to 0 (zero). The sequence number may cycle to 0 (zero) when it increases beyond its maximum defined value.

4.4. Bootstrapping of Additional Mobile IPv6 Parameters

When the MN contacts the HAC to distribute of the security related information, the HAC may also provision the MN with various Mobile IPv6 related bootstrapping information. Bootstrapping of the following information SHOULD at least be possible:

Home Agent IP Address:

Concerns both IPv6 and IPv4 home agent addresses.

Home Address:

Concerns both IPv6 and IPv4 Home Addresses.

Home Link Prefix:

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

The Mobile IPv6 related bootstrapping information is delivered from the HAC to the MN over the same TLS protected tunnel as the security related information.

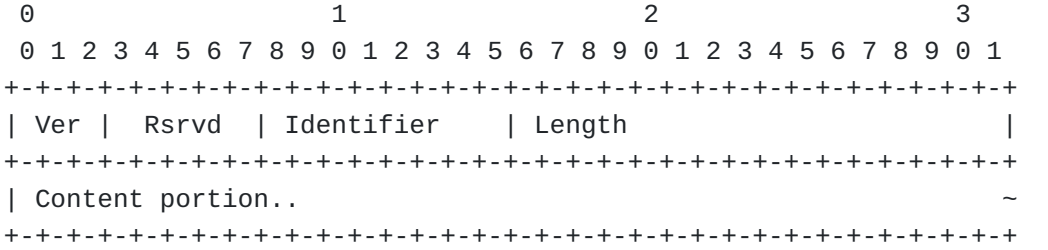
4.5. Protecting Traffic Between Mobile Node and Home Agent

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 both integrity and SHOULD be confidentially protected. The reverse tunneled user data traffic SHOULD be equivalently protected. Generally, the rules stated in [RFC3775] concerning the protection of the traffic between the MN and the HA apply also in this specification.

5. Mobile Node to Home Agent Controller Communication

5.1. Request-response Message Framing over TLS-tunnel

The MN and the HAC communicate with each other using a simple lock-step request-response protocol that is run directly on top of the TLS-tunnel. We define only one message container framing for the request messages and for the response messages. The message containers are only meant to be exchanged on top of 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 1.



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.
The three bit Ver field identifies the protocol version. The current version is 0 i.e. all bits are set to value 0 (zero).
The Rsrvd field MUST be set to value 0 (zero),
The Identifier field is meant for matching 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 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 to 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 TypeValue header (TV-header) term to refer 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 lock-step request-response type as stated in Section 5.1. A request message includes monotonically increasing Identifier value that is copied to the corresponding response message. Each request MUST have a different Identifier value and due the assumption of a reliable connection oriented transport below the message container framing. 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 a case of a HAC side error, the HAC MUST send a response back to a 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.

And in a 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 Mobile IPv6 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 to the MN or dynamically discovered using for example DNS. In a case of DNS-based HAC discovery, the MN either queries 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

5.5.1. Mobile Node Identifier

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

```
*mn-id = "mn-id" ":" nai CRLF
 nai = username
     | "@" realm
     | username "@" realm
 ...
```

5.5.2. Authentication Method

The HAC is the peer that mandates the used authentication method. The MN sends its proposal to the HAC but eventually the used authentication method returned from the HAC defines the one to be used. The MN MUST propose at least one authentication method and it SHOULD propose more than one. 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" ; Pre-sharer key based authentication
    | "eap" ; EAP-based authentication
```

5.5.3. Extensible Authentication Protocol Payload

Each Extensible Authentication Protocol (EAP) [RFC3748] message is encoded string of hexadecimal numbers. The "eap-payload" is completely transparent what 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

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

5.5.7. End of Message Content

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

5.5.8. Random Values

Random number generated by the MN or the HAC. 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 v1.2 (see Annex A.5 of [RFC5246]) as 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 confidentially 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, [RFC3268] 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 affecting to 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:

```
*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:

```
*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:

```
*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 parameter MUST be agreed between the MN and HA prior to using the SA. Otherwise the receiving side would not be aware of whether the SA applies to data traffic and could not 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 with PType = 0 MUST be silently discarded when received.
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 Annex A.5 of [RFC5246]. The TV-headers corresponding to the selected ciphersuite and ciphersuite list are:

```
*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 and subject to future specifications.

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

The following integrity algorithms MUST be supported by all implementations:
The binding management messages between the MN and HA MUST be integrity protected. Implementations MUST NOT use a NULL integrity algorithm.

NULL	[RFC2410]	TripleDES-CBC	[RFC2451]

The following encryption algorithms MUST be supported:
Traffic between MN and HA MAY be encrypted. Any integrity-only CipherSuite makes use of the NULL encryption algorithm.

Ciphersuite	Integ
NULL_SHA	HMAC-S

Note: In the present version, 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.

5.7. Mobile IPv6 Bootstrapping Parameters

In parallel with the SA bootstrapping, the HAC SHOULD provision the MN with relevant Mobile IPv6 related bootstrapping information.

```
   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
```

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

5.7.1. Home Agent Address

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

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

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

5.7.2. Home Addresses and Home Network Prefix

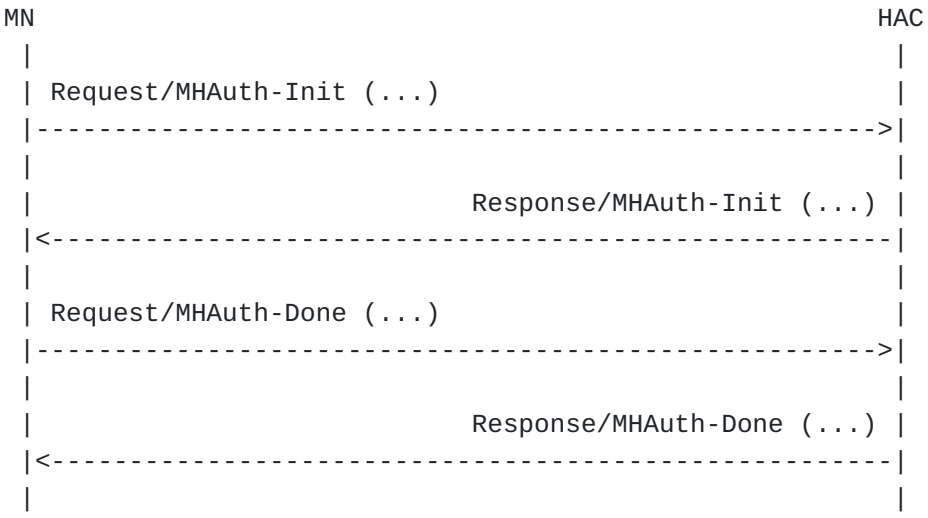
The HAC MAY provision the MN with an IPv4 or an IPv6 home address, or both. The HAC MAY also provision the MN with its home network prefix.

```
*mip6-ip6-hoa = "mip6-ip6-hoa" ":" ip6-addr CRLF

*mip6-ip4-hoa = "mip6-ip4-hoa" ":" ip4-addr CRLF

*mip6-hnp-ip6 = "mip6-ip6-hnp" ":" ip6-prefix 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 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 [I-D.altman-tls-channel-bindings]. 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 pre-shared key (PSK) or based on an EAP-method (see Section 5.9). 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 [I-D.altman-tls-channel-bindings] 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 6 shows a successful protocol exchange.



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

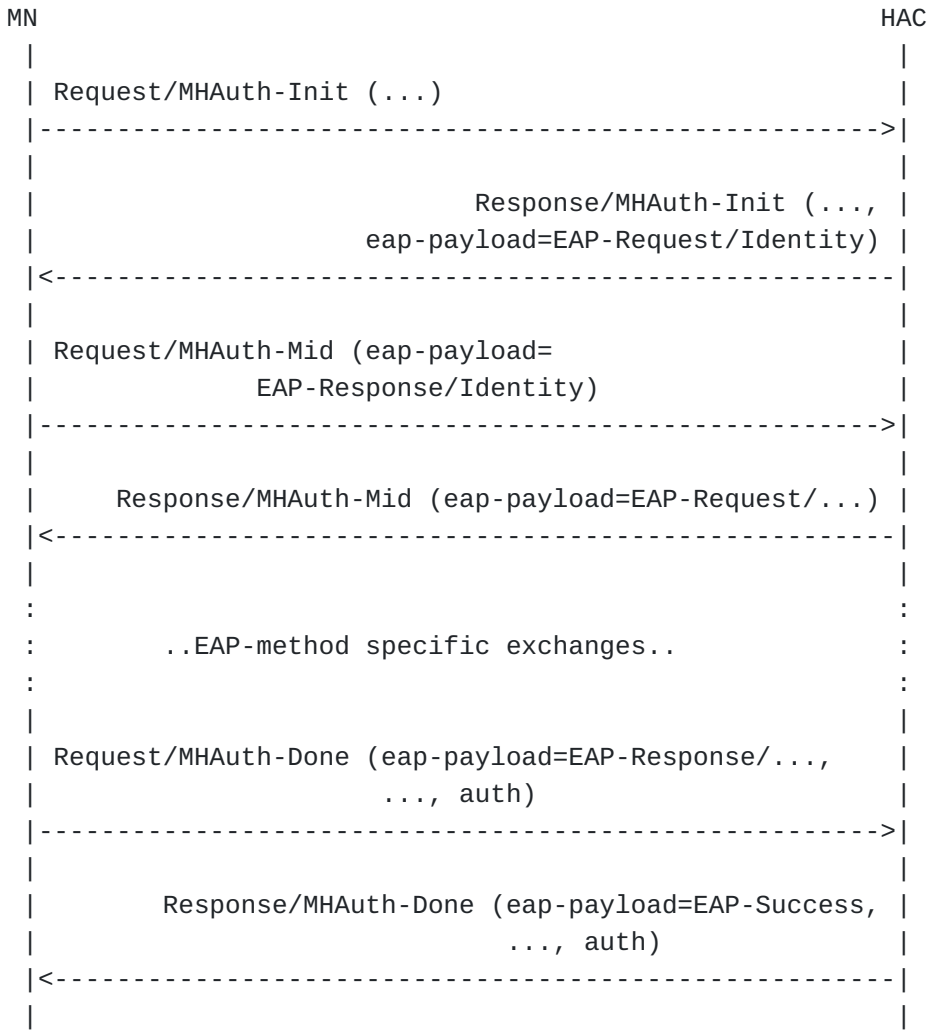
Where:

*auth = HMAC-SHA256(PSK, msg-octets | CB-octets)

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 [I-D.altman-tls-channel-bindings].
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 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.



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

In MHAuth-Init and MHAuth-Mid messages, shared-key is set to "1". If the EAP-method is key-deriving and creates a shared MSK key 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 key, 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 [I-D.altman-tls-channel-bindings].

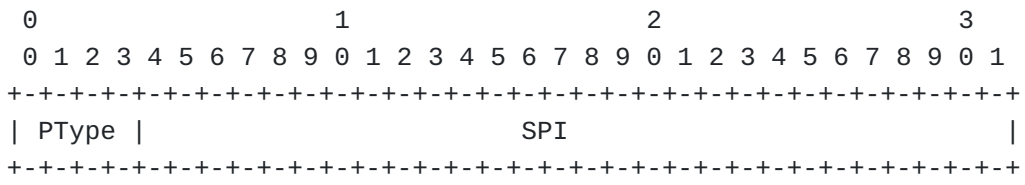
6. Mobile Node to Home Agent communication

6.1. General

The following sections 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 plain text user data. This specification defines a generic packet format, where everything is encapsulated inside UDP. See Section 6.3 and Section 6.4 for detailed illustrations of the corresponding packet formats.

The Mobile IPv6 service port number (HALTSEC), where the HA expects to receive UDP packets, is reserved by IANA. 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 MUST use the Mobile IPv6 service port number (HALTSEC) as the UDP destination port.
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 or a, an encrypted user data packet, or a plain text 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 a 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 encrypted messages. 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 a case encryption is not used, the Sequence Number MUST be set to 0 (zero). Note that the HA SHOULD initiate a re-establishement 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. Security Parameter Index



The SPI is a 32-bit field, where the first 4 bits indicate the Packet Type (PType) of the UDP encapsulated packet. The SPI value itself consists of the remaining 28-bit of the SPI field. The SPI field is treated as one 32-bit field during the integrity protection calculation.

6.3. Binding Management Message Formats

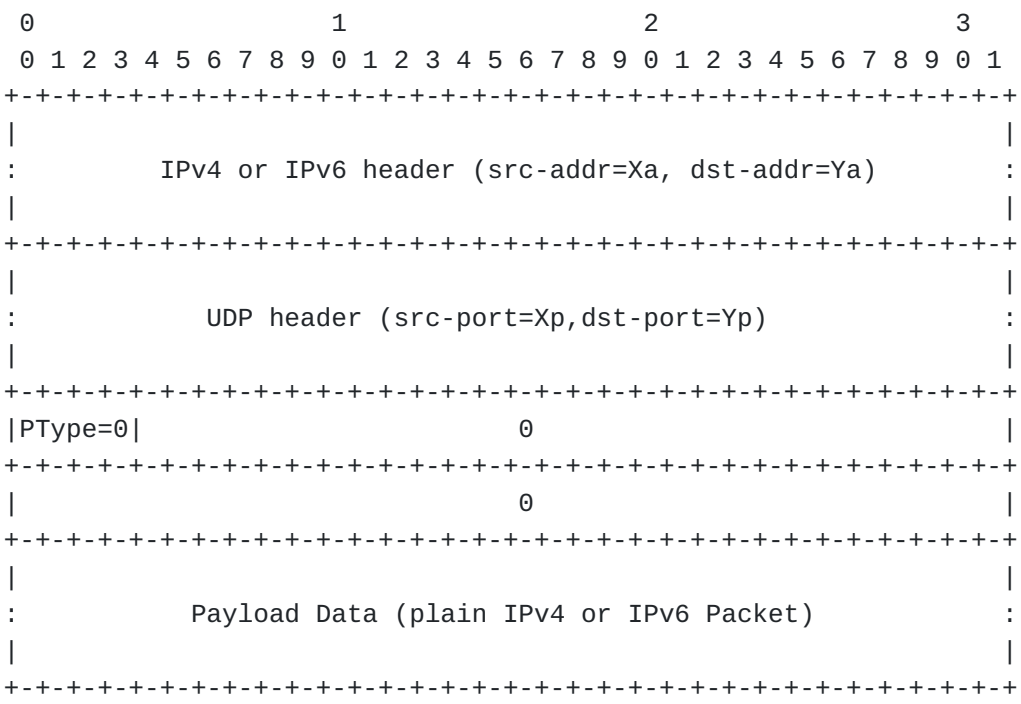
The binding management messages that are only meant to be exchanged between the MN and the HA MUST be integrity protected and MAY be encrypted. They MUST use the packet format shown in Figure 9. All packets that are specific to the Mobile IPv6 protocol and contain a Mobility Header (as defined in Section 6.1.1. of RFC 3775) SHOULD use the packet format shown in Figure 9. (This means that some Mobile IPv6 mobility management messages, such as the HoTI message, as treated as data packets and using encapsulation described in Section 6.4).

The PType value 8 (eight) identifies that the UDP encapsulated packet contains a RFC 3775 defined Mobility Header 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 a 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 9) use the current care-of address 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 encrypted and those that are plaintext. The MN or the HA decide whether to apply integrity protection and encryption to a packet or to send it in plaintext based on the mip6-sas value in the SA. If the mip6-sas is set to 1 the originator MUST NOT send any plaintext packet, and the receiver MUST silently discard any packet with the PType set to 0 (unprotected). It is RECOMMENDED to apply confidentiality and integrity protection of user data traffic. The reverse tunneled IPv4 or IPv6 user data packets are encapsulated as-is inside the 'Payload Data' shown in Figure 10. and Figure 11.
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 a 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 10) use the current care-of address of the MN and the HA address. The ESP protected inner IP header, which is not shown in Figure 10, uses the home address of the MN and the correspondent node (CN) address.



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 11) use the current care-of address of the MN and the HA address. The plain text inner IP header uses the home address of the MN and the CN address.

7. Route Optimization

The treatment of MN-CN route optimization is outside the scope of this document.

8. IANA Considerations

8.1. New Registry: Packet Type

Packet Type	Value
non-encrypted IP packet	0
encrypted IP packet	1
mobility header	8

IANA is requested to create a new registry for the Packet Type as described in Section 6.1.
Following the allocation policies from [RFC5226] new values for the Packet Type AVP MUST be assigned based on the "RFC Required" policy.

8.2. HTTP Headers

A number of HTTP headers with their respective parameters are reserved. See Section 5.6 and Section 5.7 for a list of header names and their parameters.

8.3. Status Codes

REINIT_SA_WITH_HAC	TBD1

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 Result Code is reserved from the 0-127 code space:

8.4. Port Numbers

HALTSEC	TBD2

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

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 1.2 [RFC5246].

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 section 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 either be 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 key in compliance with [RFC3748].

Both exchanges use channel binding with the TLS tunnel. The channel binding type ‘TLS-server-endpoint’ as per [I-D.altman-tls-channel-bindings] 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 as per section 5.8. is used, and are the same as those of the underlying EAP method in case the EAP-based exchange as 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 Resistance: The level of resistance against denial of service 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 denial-of-service resistance specific to the chosen EAP method are expected to be applicable and need to be 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' as per [I-D.altman-tls-channel-bindings] 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 RFC 4086 [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 (in-bound communication or out-bound communication).

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

Denial of Service 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 by 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 a case binding management messages contain the user identity, the messages SHOULD be confidentity 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 therefore out-of-scope.

10. Acknowledgements

The authors would like to thank Pasi Eronen, Domagoj Premec, 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.
[RFC3775]	Johnson, D., Perkins, C. and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.
[RFC5246]	Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
[RFC5226]	Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
[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.
[RFC5056]	Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, November 2007.
[RFC4282]	Aboba, B., Beadles, M., Arkko, J. and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.
[RFC2404]	Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.
[RFC3566]	Frankel, S. and H. Herbert, "The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec", RFC 3566, September 2003.
[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.
[RFC3602]	Frankel, S., Glenn, R. and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, September 2003.
[RFC4285]	Patel, A., Leung, K., Khalil, M., Akhtar, H. and K. Chowdhury, "Authentication Protocol for Mobile IPv6", RFC 4285, January 2006.
[I-D.altman-tls-channel-bindings]	Altman, J, Williams, N and L Zhu, "Channel Bindings for TLS", Internet-Draft draft-altman-tls-channel-bindings-10, March 2010.

11.2. Informative References

[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.
[RFC4306]	Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.
[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.
[RFC4877]	Devarapalli, V. and F. Dupont, "Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture", RFC 4877, April 2007.
[RFC3344]	Perkins, C., "IP Mobility Support for IPv4", RFC 3344, August 2002.
[RFC5555]	Soliman, H., "Mobile IPv6 Support for Dual Stack Hosts and Routers", RFC 5555, June 2009.
[RFC3268]	Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.
[RFC4279]	Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.
[RFC4086]	Eastlake, D., Schiller, J. and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.
[RFC5077]	Salowey, J., Zhou, H., Eronen, P. and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.
[RFC3748]	Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J. and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.
[RFC5433]	Clancy, T. and H. Tschofenig, "Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK) Method", RFC 5433, February 2009.

Authors' Addresses

Basavaraj Patil Patil Nokia 6021 Connection Drive Irving,, TX 75039 USA EMail: basavaraj.patil@nokia.com

Charles Perkins Perkins Tellabs 3590 N. 1st Street, Suite 300 San Jose,, CA 95134 USA EMail: charles.perkins@tellabs.com

Hannes Tschofenig Tschofenig Nokia Siemens Networks Linnoitustie 6 Espoo, 02600 Finland Phone: +358 (50) 4871445 EMail: Hannes.Tschofenig@gmx.net URI: http://www.tschofenig.priv.at

Domagoj Premec Premec Unaffiliated Heinzelova 70a Zagreb, 10000 Croatia EMail: domagoj.premec@gmail.com