EAP WG
Internet-Draft

H. Tschofenig
D. Kroeselberg
Siemens
Y. Ohba
Toshiba
F. Bersani
France Telecom R&D

Document: draft-tschofenig-eap-ikev2-07.txt
Expires: January 18, 2006 July 2005

EAP IKEv2 Method (EAP-IKEv2)

Status of this Memo

Copyright Notice

Abstract

EAP-IKEv2 is an EAP authentication method with cryptographic properties and basic exchanges similar to the Internet Key Exchange (IKEv2) protocol. It provides a flexible EAP method with support for both symmetric and asymmetric authentication, as well as a combination of both.
EAP-IKEv2 thereby offers the security benefits of the exchanges for authentication and key agreement of IKEv2 within the AAA framework defined by the IETF.

Table of Contents

1. Introduction

This document specifies the EAP authentication method EAP-IKEv2. The main design goal for EAP-IKEv2 is to provide a flexible and efficient EAP method which makes security properties and exchanges similar to these of the IKEv2 protocol available for all scenarios using EAP-based authentication.
The main advantage of EAP-IKEv2 is that it does not define a new cryptographic protocol, but re-uses the well-analyzed IKEv2 authentication exchanges within the EAP framework. Thereby, it provides strong cryptographic properties as well as good flexibility to support a large number of use cases.

EAP-IKEv2 especially provides an efficient shared-secret based authentication method offering a high security level, and allows

for password-derived shared secrets while protecting from password-guessing attacks.

It provides mutual authentication between EAP peers. This may be based on either symmetric-key methods using pre-shared keys, or on asymmetric methods based on public/private key pairs, Certificates and CRLs. It is possible to use different types of authentication for the different directions, e.g. the server uses certificate-based authentication whereas the client uses a symmetric-key method.
By this, both AAA scenarios where public-key EAP-based authentication as well as scenarios requiring symmetric-key EAP-based authentication are flexibly supported.

2. EAP-IKEv2 Overview

EAP-IKEv2 is an EAP authentication method that offers security features similar to those offered by the IKEv2 protocol defined for Internet key exchange. It defines exchanges and message formats similar to exchanges and payloads specified by IKEv2 for establishment of an IKE-SA.

The basic successful EAP-IKEv2 exchange as specified in section 4 requires two roundtrips for authenticating EAP peer and server that are followed by an EAP-Success message. An optional roundtrip for exchanging EAP identities may precede the authentication exchange.
In addition to the basic exchange, a fast reconnect method is specified in section 11 to allow fast session resumption with increased efficiency compared to an EAP-IKEv2 standard exchange.

Section 5 details the handling of identities for EAP-IKEv2, since identities occur in both the basic EAP exchange as well as the specific EAP-IKEv2 authentication exchange.

In section 6, the packet format for EAP-IKEv2 messages is specified, which is composed of the standard EAP request/response message and the EAP method specific formats that are derived from the original IKEv2 protocol specification.

EAP-IKEv2 provides a secure fragmentation mechanism that is detailed in section 7 and details retransmission aspects in section 8.

Key derivation as an important part of any EAP authentication method is specified in section 9 that details the method-specific behavior according to the overall EAP keying framework.

For security aspects, in section 12 a detailed discussion on channel binding to avoid security issues related to misbehaving EAP authenticators can be found. The general security considerations for this EAP method are subsequently given in section 13.

In general, although EAP-IKEv2 reuses parts of the original IKEv2 specification, it must be noted that the scenarios EAP-IKEv2 is intended for are clearly different from the scenarios covered by IKEv2. Therefore, a number of mechanisms available in IKEv2 are not required, nor are they available, in EAP-IKEv2. For example, the optional tunneling of IKEv2 (inner authentication method as defined in [Kau04], section 3.16) is not supported by this version of EAP-IKEv2.

EAP-IKEv2 provides authentication between an EAP server and an EAP peer in a single authentication exchange, or phase. In contrast, IKEv2 [Kau04] itself is a protocol which consists of two phases that can be run (but are not necessarily run) subsequently:

(1) an authentication and key exchange phase which establishes an IKE-SA.

(2) a phase for the negotiation of parameters in order to establish IPsec security associations. Such exchanges contain e.g. algorithm parameters and traffic selector fields, and are protected by the security established in the first phase.

The EAP-IKEv2 method does not include any exchange similar to the above phase (2), since such functionality is not a requirement in the context of common AAA scenarios, consisting of an EAP peer, an authenticator (NAS) and a back-end authentication server. There, IPsec SA establishment may be required locally (i.e., between the EAP peer and some access server). However, SA establishment within an EAP method could only provide SAs between

the EAP peer and the back-end authentication server. Other approaches as, e.g., the IETF PANA framework are considered more appropriate in this case.

3. Terminology

This document does not introduce new terms other than those defined in [RFC3748] or in [Kau04].

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

4. Protocol overview

This section defines the basic EAP-IKEv2 message exchanges.

The given exchanges are based on IKEv2 [Kau04].

All message exchanges take place between two parties - between the Initiator (I) and the Responder (R) of an exchange. In the context of EAP the Initiator takes the role of the EAP server and the responder matches the EAP peer.

The first message flow shows the EAP-IKEv2 full successful authentication exchange. The core EAP-IKEv2 exchange (message (3) to (6)) consists of four messages (two round trips)_only:

- Messages (3) and (4) negotiate cryptographic algorithms, exchange nonces, and perform a Diffie-Hellman exchange. This step is called EAP-IKE_SA_INIT exchange.
- Messages (5) and (6) authenticate the EAP-IKE_SA_INIT exchange, and exchange the identities of Initiator and Responder (i.e. the EAP server and peer) and certificates. This step is called EAP-IKE_SA_AUTH exchange.

The first two messages (1) and (2) constitute the standard EAP identity exchange and are optional; they are not required in case the EAP server is known. The exchange is concluded with an EAP-Success message (7) sent by the EAP server to the EAP peer.

In the exchange, the EAP server (B) takes the role of the Initiator and the EAP peer (A) acts as the Responder.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SAr1, KEr, Nr, [CERTREQ])

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH})
```



```
6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDr, [CERT,] AUTH})

7) A <-- B: EAP-Success
```

Figure 1: EAP-IKEv2 successful message flow

Descriptions of the EAP-IKEv2 message format, headers and payloads are given in section 6.

Figure 2 shows the message flow in case the EAP peer fails to authenticate the EAP server. The difference to the above successful exchange is that in message (6) the EAP peer answers to the EAP server with an AUTHENTICATION_FAILED error. In message (7), EAP-Failure is returned from the EAP server.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SAr1, KEr, Nr, [CERTREQ])

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH})
```

```
6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(AUTHENTICATION_FAILED)})

7) A <-- B: EAP-Failure
```

Figure 2: EAP-IKEv2 with failed server authentication

Figure 3 shows the message flow in case the EAP server fails to authenticate the EAP peer. Compared to the successful exchange, one additional roundtrip is required. In message (7) the EAP server answers with an AUTHENTICATION_FAILED error instead of sending EAP-Successs. The EAP peer, after receiving message (7), MUST send an empty EAP-IKEv2 (informational) message in reply to the EAP server's error indication, as shown in (8). The EAP server answers with an EAP-Failure.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)
```



```
3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SAr1, KEr, Nr, [CERTREQ])

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDr, [CERT,] AUTH})

7) A <-- B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(AUTHENTICATION_FAILED)})

8) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {})

9) A <-- B: EAP-Failure
```

Figure 3: EAP-IKEv2 with failed client authentication

Since the goal of this EAP method is not the goal of the original IKEv2 protocol to establish IPsec security associations, some

payloads that are specified in IKEv2 are not specified for EAP-IKEv2 (as they do not occur in the message exchanges specified in this document). For example, the following messages and payloads are not specified for EAP-IKEv2::

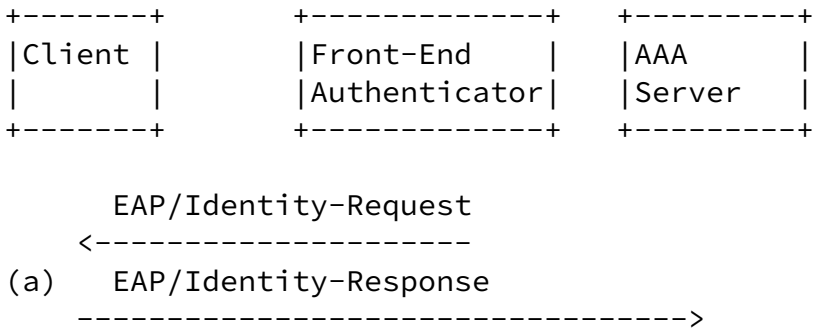
- Traffic Selector (TS) payloads ([Kau04], section 3.13).
- SA payloads ([Kau04], section 3.3) that carry SA proposals for protocol IDs other than 1(IKE), i.e., SA payloads with protocol ID 2 (ESP) or 3 (AH)
- Configuration payloads ([Kau04], section 3.15).
- EAP payloads ([Kau04], section 3.16), since EAP tunnelling within EAP-IKEv2 is not supported in this version of EAP-IKEv2.

Consequently, mechanisms that are part of IKEv2 but are not required nor specified within EAP-IKEv2 are:
- ECN Notifications as specified in section 2.24 of [Kau04].
- IKE-specific port handling
- NAT traversal

IKEv2 provides optional functionality for additional DoS protection by adding a roundtrip to the initial exchanges, see section 2.6 of [Kau04]. This is intended to protect the IKEv2 responder. Because in EAP-IKEv2 the EAP server takes the role of the initiator, no similar feature is specified for EAP-IKEv2.

5. Identities used in EAP-IKEv2

A number of different places allow to convey identity information in IKEv2 as well as in EAP. This section describes identities and their role within the different exchanges of EAP-IKEv2. Note that EAP-IKEv2 does not introduce more identities than other non-tunneling EAP methods. Figure 4 shows which identities are used during the individual phases of the protocol.



```
        EAP-IKE_SA_INIT and AUTH exchange
 (b)    (Identities of IKEv2 are used)
        Server (Network) Authentication
     <-----------------------------------
                   ...
     ----------------------------------->
```

Figure 4: Identities used in EAP-IKEv2

a) The first part of the EAP message exchange provides information about the identities of the EAP endpoints. This message exchange mainly is an identity request/response. It is optional if the EAP server is known already or can be learned by other means.

b) Identities IDi and IDr are exchanged within the EAP-IKE_SA_INIT and AUTH exchanges for both the initiator and the responder (EAP server and peer).

For carrying identities in EAP-IKEv2, implementations MUST follow the rules given in [Kau04], section 3.5, i.e., MUST be configurable to send at least one of ID_IPV4_ADDR, ID_FQDN, ID_RFC822_ADDR, or ID_KEY_ID, and MUST be configurable to accept all of these types. Implementations SHOULD be capable of generating and accepting all of these types.

6. Packet Format

The EAP-IKEv2 messages are EAP messages carrying the authentication exchange embedded in the Data field of the standard EAP Request/Response packets. The Code, Identifier, Length and Type fields are described in [RFC3748]. These are followed by a Type-Data field that carries one octet with method-specific Flags as specified in section 7.
This EAP header is, embedded in the Data field, followed by the method-specific header HDR and by one or more payloads of the EAP-IKEv2 authentication data.

The EAP Type for this EAP method is <TBD>.

The general packet format is shown in Figure 5.

```
   0                   1                   2                   3
```

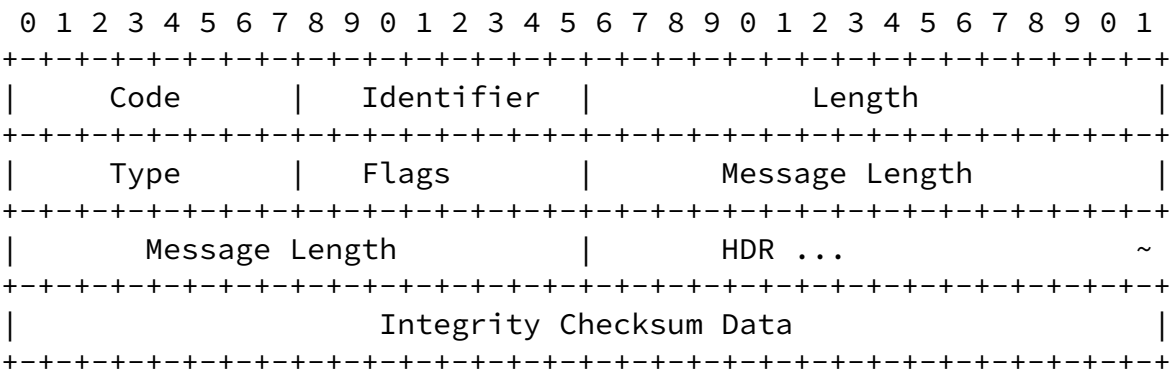


Figure 5: EAP-IKEv2 general packet format

The HDR payload that heads the Data field is as shown in Figure 5Figure 6 below. The HDR fields given in Figure 6 are used according to [Kau04], section 3.1, where the EAP server acts as the initiator and the EAP peer acts as the responder.

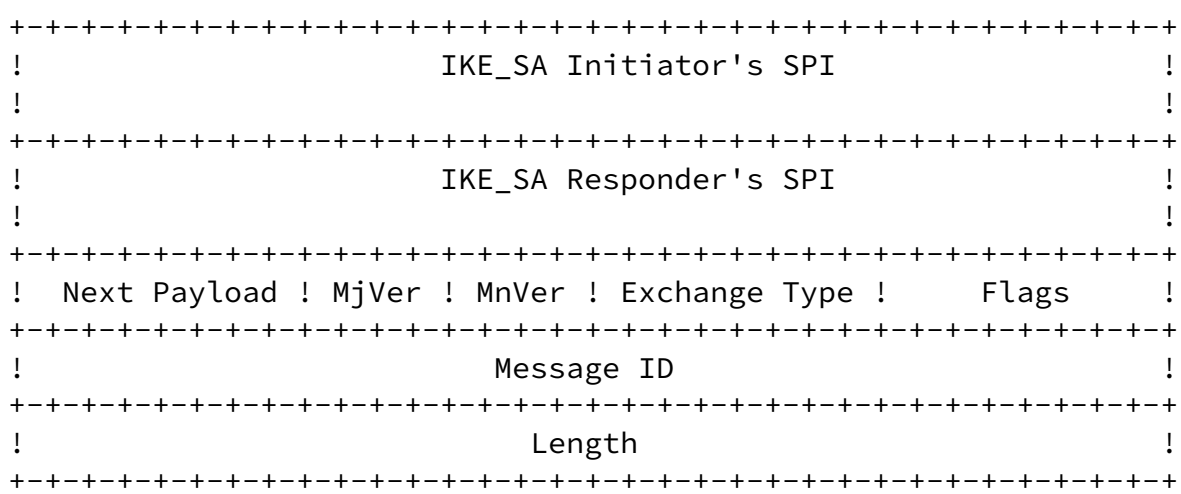


Figure 6: HDR format

In contrast to the original IKEv2 protocol specified in [Kau04], the HDR (IKE header data) is not carried within a UDP message, but is directly embedded into the EAP message as shown above. However, no additional packet formats other than those defined in [Kau04] are required for this EAP method.

Following the header HDR are one or more payloads where each of them is identified by a "Next Payload" field in the preceding payload. Processing these payloads follows the rules specified by [Kau04] section 3.2.
Each payload begins with a generic payload header as given in

Figure 7 followed by the payload data itself.

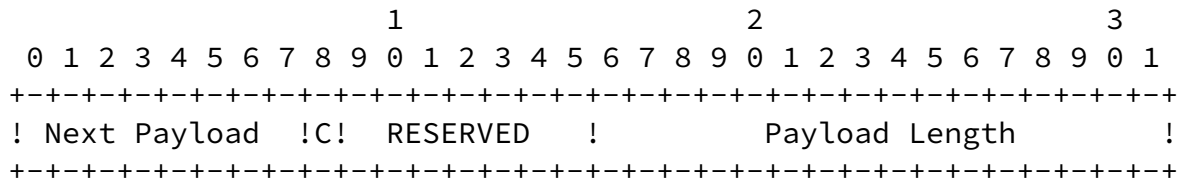


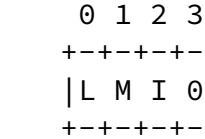
Figure 7: generic payload header format

All payloads used within the EAP-IKEv2 messages defined by this document are specified in [Kau04], sections 3.3. to 3.12 and 3.14.

7. Fragmentation support

The Flags field in HDR (see section 6) is used for fragmentation support. The S and F bits are reserved for future use.

Currently three bits of the eight bit flags field are defined. The remaining bits are set to zero.



L = Length included
M = More fragments
I = Integrity Checksum Data included

With regard to fragmentation we adopt the mechanism given in Section 2.8 of [PS+03]: The L indicates that a length field is present and the M flag indicates fragments. The L flag MUST be set

for the first fragment and the M flag MUST be set on all fragments expect for the last one.
Reliable fragment delivery is provided by the retransmission mechanism of EAP.

The Message Length field is four octets long and present only if the L bit is set. This field provides the total message length that is being fragmented (i.e., the length of the Data field.).

The Integrity Checksum Data is the cryptographic checksum of the entire EAP message starting with the Code field through the Data field. This field presents only if the I bit is set. The field immediately follows the Data field without adding any padding octet before or after itself. The checksum MUST be computed for each fragment (including the case where the entire IKEv2 message is carried in a single fragment) by using the same key (i.e., SK_ai or SK_ar) that is used for computing the checksum for the IKEv2 Encrypted payload in the encapsulated IKEv2 message. The Integrity Checksum Data field is omitted for other packets. To minimize DoS attacks on fragmented packets, messages that are not protected SHOULD NOT be fragmented.
Note that EAP-IKE_SA_INIT messages are not encrypted or integrity protected. However, they are not likely to be fragmented since they do not carry certificates.

8. Retransmission

Since EAP authenticators support a timer-based retransmission mechanism for EAP Requests and EAP peers retransmit the last valid EAP Response when receiving a duplicate EAP Request message, IKEv2 messages MUST NOT be retransmitted based on timers, when used as EAP authentication method.

9. Key derivation

The EAP-IKEv2 method described in this document generates session keys. On the one hand, these session keys are used within the IKE-SA, for protection of EAP-IKEv2 payloads, e.g., AUTH exchanges or notifications. On the other hand, additional keys are derived to be exported as part of the EAP keying framework [AS+05] (i.e., MSK, EMSK and IV).
For key derivation, EAP-IKEv2 reuses the key derivation function as specified in Section 2.17 of [Kau04] to create keying material KEYMAT.

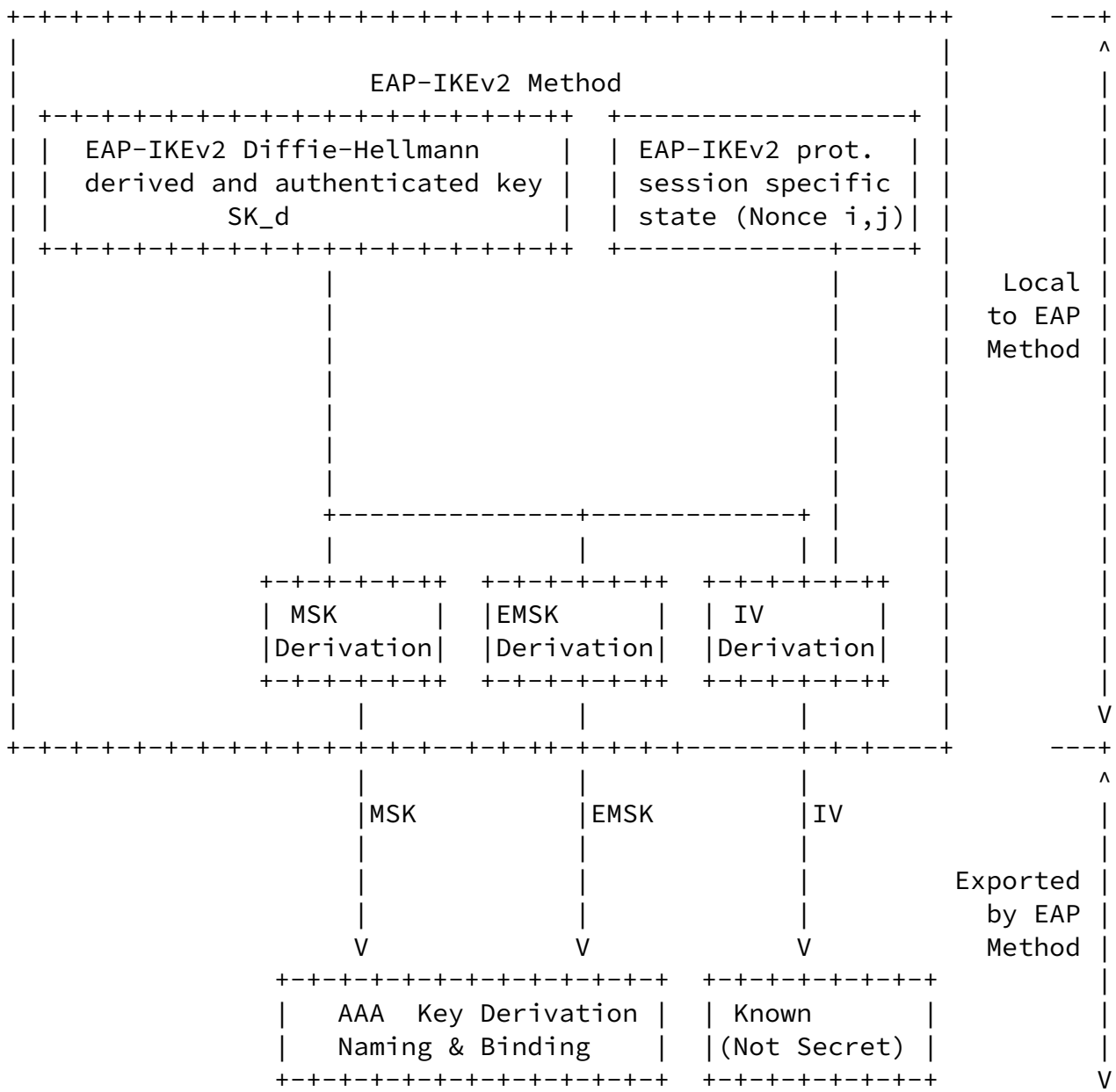
The key derivation function defined is KEYMAT = prf+(SK_d, Ni | Nr), where Ni and Nr are the Nonces from the EAP-IKE_SA_INIT exchange.

Since the required amount of keying material is greater than the

size of the output of the prf algorithm the prf is used iteratively. Iteration is applied as specified in section 2.13 of [Kau04].

The produced keying material for MSK, EMSK and IV MUST be 64 octets long for each MSK, EMSK and IV.

Figure 8 describes the keying hierarchy of EAP-IKEv2 graphically. This figure is adopted from Figure 2 of [AS+05].



Legend:

```
MSK = Master Session Key (512 Bit)
EMSK = Extended Master Session Key (512 Bit)
SK_d = Session key derived by EAP-IKEv2
IV   = Initialization Vector
```

Figure 8: EAP-IKEv2 Keying Hierarchy

10. Error Handling

As described in the IKEv2 specification, there are many kinds of errors that can occur during IKE processing (i.e., processing the Data field of EAP-IKEv2 Request and Response messages) and detailed processing rules. EAP-IKEv2 follows the error handling rules specified in the IKEv2 specification for errors on the Data field of EAP-IKEv2 messages, with the following additional rules:

For an IKEv2 error that triggers an initiation of an IKEv2 exchange (i.e., an INFORMATIONAL exchange), an EAP-IKEv2 message that contains the IKEv2 request that is generated for the IKEv2 exchange MUST be sent to the peering entity. In this case, the EAP message that caused the IKEv2 error MUST be treated as a valid EAP message.

For an IKEv2 error for which the IKEv2 message that caused the error is discarded without triggering an initiation of an IKEv2 exchange, the EAP message that carries the erroneous IKEv2 message MUST be treated as an invalid EAP message and discarded as if it were not received at EAP layer.

For an error occurred outside the Data field of EAP-IKEv2 messages, including defragmentation failures, integrity check failures, errors in Flag and Message Length fields, the EAP message that caused the error MUST be treated as an invalid EAP message and discarded as if it were not received at EAP layer.

When the EAP-IKEv2 method runs on a backend EAP server, the error handling rules defined in Section 2.2 of [RFC3579] are applied for invalid EAP-IKEv2 messages.

11. Fast Reconnect

EAP-IKEv2 supports fast reconnect, i.e., a successful reconnect exchange creates a new IKE-SA by using a method similar to the IKE CHILD_SA exchange defined in [KAU04]. The purpose of a re-authentication exchange is to allow for efficient re-keying based on the existing IKE-SA in situations where (depending on the

given security policy) no full authentication is required in case of an existing EAP-IKEv2 security context.

The fast reconnect exchange is similar to the IKE-SA rekeying procedure as specified in section 2.18 of [Kau04]. However, the exchanges for EAP-IKEv2 that are specified below do not use rekeying payloads other than IKE SAs:

- The TS (traffic selector) payloads are not used in EAP-IKEv2.
- The [N] payload (REKEY_SA notification) is not sent by EAP-IKEv2.

During fast re-authentication, the new IKE_SA is computed as specified in [Kau04], section 2.18. The new keying material derived from this IKE_SA is computed in the same way as in an initial EAP-IKEv2 authentication exchange.
Fast re-authentication allows for an optional fresh Diffie-Hellman exchange in case the payloads Kei and KEr are included.

The following exchanges specify fast reconnect for EAP-IKEv2, where A is the EAP peer (responder) and B is the EAP server (initiator):

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
            HDR, SK {SA, Ni, [KEi]})

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
            HDR, SK {SA, Nr, [KEr]})

5) A <-- B: EAP-Success
```

Figure 9: Fast Reconnect exchange

The first two messages constitute the standard EAP identity exchange and are optional; they are not required in case the EAP server is known. Messages (3) and (4) establish the new IKE SA. The successful fast reconnect is concluded by an EAP-Success sent by the EAP server.

Figure 10 shows the fast reconnect message flow in case the EAP peer fails to re-authenticate the EAP server.

```
1) A <-- B: EAP-Request/Identity
```

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2
(HDR, SK {SA, Ni, [KEi]})

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
HDR, SK {N(AUTHENTICATION_FAILED)})

5) A <-- B: EAP-Failure

Figure 10: EAP-IKEv2 fast reconnect
(server authentication failed)

Figure 11 shows the fast reconnect message flow in case the EAP server fails to re-authenticate the EAP peer. The EAP peer MUST send an empty EAP-IKEv2 informational message (empty encrypted payload) in reply to the EAP server's error indication, as shown in (6) below. The EAP server answers with an EAP-Failure.

1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
HDR, SK {SA, Ni, [KEi]})

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
HDR, SK {SA, Nr, [KEr]})

5) A <-- B: EAP-Response/EAP-Type=EAP-IKEv2(
HDR(A,B), SK {N(AUTHENTICATION_FAILED)})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
HDR(A,B), SK {})

7) A <-- B: EAP-Failure

Figure 11: EAP-IKEv2 fast reconnect
(client authentication failed)

Note: The original IKEv2 protocol supports fast rekeying to be initiated by both peers. IKE_SAs do not have lifetimes. Such lifetimes are therefore set by local policies. Typically the peer

setting the shorter lifetime will therefore trigger the reconnect procedure in IKEv2.
In EAP-IKEv2, the EAP authenticator or server initiate the rekeying as this results in the most efficient message flow. If

the client initiates fast rekeying, it needs to indicate this to the network by appropriate out-of-band (e.g. link-layer) means.

12. Channel Binding

EAP-IKEv2 provides a channel binding functionality [RFC3784] in order for the EAP peer and EAP server to make sure that the both entities are given the same network access attributes such as Calling-Station-Id, Called-Station-Id, and NAS-Port-Type by the NAS. This is achieved by using Notify payloads to exchange attribute data between the EAP peer and EAP server.

A Notify payload that carries a null channel binding attribute is referred to as a channel binding request. A Notify payload which contains a non-null channel binding attribute and is sent in response to a channel binding request is referred to as a channel binding response. A pair of channel binding request and channel binding response constitutes a channel binding exchange. A distinct Notify payload type is used for a particular type of channel binding attribute, which is referred to as a channel binding attribute type. It is allowed to carry multiple channel binding requests and/or responses with different channel binding attribute types in a single IKEv2 message. A set of channel binding exchanges performed in a single round of EAP-IKEv2 full authentication or fast reconnect is referred to as a channel binding procedure.

A Notify payload that is used for reporting an error occurred during a channel binding exchange is referred to as a channel binding error indication.

EAP-IKEv2 offers a protected result indication mechanism (see section 13.2). To receive protected result indication, the EAP server MUST initiate a channel binding exchange as specified in Figure 12, message 5. As a result of this channel binding exchange, the client will receive a protected result indication, because the server will initiate an informational exchange as part of the channel binding procedure (messages 7 and 8) through the new IKE-SA that results from a successful reconnect procedure.

12.1 Channel Binding Procedure in Full Authentication

In the case of EAP-IKEv2 full authentication procedure, the channel binding procedure is performed in the following way.

The EAP peer MAY include one or more channel binding request in an IKE_SA_INIT response. The EAP server MAY include one or more channel binding request in an IKE_AUTH request. When the EAP server

receives an IKE_SA_INIT response with one or more channel binding request, it MUST include the corresponding channel binding response(s) an IKE_AUTH request (in addition to its channel binding request(s) if any). When the EAP peer receives an IKE_AUTH request with one or more channel binding request, it MUST include the corresponding channel binding response(s) in an IKE_AUTH response.

When the EAP server successfully validates all the channel binding response(s) sent by the EAP server, it initiates an INFORMATIONAL exchange, where an empty payload is used in both INFORMATIONAL request and INFORMATIONAL response. This exchange serves as a protected success indication. After completion of this INFORMATIONAL exchange, the EAP server sends Success message.

12.2 Channel Binding Procedure in Fast Reconnect

In the case of EAP-IKEv2 fast reconnect, the channel binding procedure is performed in the following way.

In the pair of CREATE_CHILD_SA exchange, the EAP peer and/or the EAP server MAY include one or more channel binding request, one for each channel binding attribute that needs validation. When the EAP peer receives a CREATE_CHILD_SA request with containing one or more channel binding request, it MUST contain channel binding response(s) in the CREATE_CHILD_SA response, as well as its channel binding request(s) if any. This piggybacking is possible because the CREATE_CHILD_SA exchange is protected with the old IKE_SA. When the EAP server receives a CREATE_CHILD_SA response, if it has one or more channel binding response to send to the EAP peer, it initiates an INFORMATIONAL exchange immediately after completion of the CREATE_CHILD_SA exchange, where one or more channel binding response is carried in the INFORMATIONAL request. If the EAP peer successfully validates the channel binding response(s), it MUST respond with an empty INFORMATIONAL response. This exchange serves as a protected

success indication. After completion of this INFORMATIONAL exchange, the EAP server sends Success message.

12.3 Channel Binding Error Indication

A channel binding error is detected by the EAP peer or EAP server when (i) a channel binding response is not contained in the expected IKEv2 message or (ii) a channel binding response is contained in the expected IKEv2 message but the channel binding attribute does not have the expected value. Whenever a channel binding error is detected, the detecting entity MUST send a channel binding error indication to its peering entity. In case of (ii),

the channel binding error indication MUST contain the channel binding attribute that caused the error.

When the EAP server detects a channel binding error, a channel binding error indication MUST be carried in an INFORMATIONAL request, and the EAP peer MUST respond with an empty INFORMATIONAL response.
When the EAP peer detects a channel binding error, a channel binding error indication MUST be carried in an IKEv2 error reporting message for which the R-flag of the IKEv2 header MUST be set. The EAP server MUST respond with EAP-Failure message when it receives such a channel binding error indication.

12.4 Notify Payload Types for Channel Binding

The following Notify Payload types are defined for the purpose of channel binding exchange.

CALLING_STATION_ID TBD
: The payload data in a channel binding response of this type contains octet string representation of Calling-Station-Id value known to the EAP server by using an external mechanism.

CALLED_STATION_ID TBD
: The payload data in a channel binding response of this type contains octet string representation of Called-Station-Id value known to the EAP peer by using an external mechanism.

NAS_PORT_TYPE TBD
: The payload data in a channel binding response of this type contains 4-octet unsigned integer value of NAS-Port-Type

known to the EAP peer by using an external mechanism.

The following Notify Payload types are defined for the purpose of reporting when there is an error in a channel binding exchange.

INVALID_CALLING_STATION_ID TBD

The payload data (if non-null) contains octet string representation of Calling-Station-Id value that caused the error.

INVALID_CALLED_STATION_ID TBD

The payload data (if non-null) contains octet string representation of Called-Station-Id value that caused the error.

INVALID_NAS_PORT_TYPE TBD

The payload data (if non-null) contains 4-octet unsigned integer value of NAS-Port-Type that caused the error.

Table 1 shows the entity that is allowed to send a channel binding request for each channel binding attribute type.

channel binding attribute type	The entity that is allowed to send channel binding request
CALLING_STATION_ID	EAP server
CALLED_STATION_ID	EAP peer
NAS_PORT_TYPE	EAP server

Table 1: Channel Binding Attribute Types and Requesting Entities

12.5 Examples

In the figures of this section, a Notify payload tagged with '*' indicates a Notify payload with null data (i.e., a channel binding request). a Notify payload no tagged with '*' indicates a Notify

payload with non-null data (i.e., a channel binding response).

Figure 12 shows an example of EAP-IKEv2 authentication sequence with a successful channel binding procedure. The first two messages constitute the standard EAP identity exchange and are optional.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SAr1, KEr, Nr, [CERTREQ,]
         N(CALLED_STATION_ID*))

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH,
         N(CALLED_STATION_ID),
         N(CALLING_STATION_ID*),
```



```
         N(NAS_PORT_TYPE*)})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDr, [CERT,] AUTH,
         N(CALLING_STATION_ID),
         N(NAS_PORT_TYPE)})

7) A <-- B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {})

8) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {})

9) A <-- B: EAP-Success
```

Figure 12: EAP-IKEv2 with successful channel binding

Figure 13 shows an example of EAP-IKEv2 authentication sequence when the EAP server detects an error in a channel binding procedure. The first two messages constitute the standard EAP identity exchange and are optional. In this case, message 7) and 8) MUST constitute an INFORMATIONAL exchange.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SAr1, KEr, Nr, [CERTREQ,]
         N(CALLED_STATION_ID*))

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH,
         N(CALLED_STATION_ID),
         N(CALLING_STATION_ID*),
         N(NAS_PORT_TYPE*)})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {IDr, [CERT,] AUTH,
         N(CALLING_STATION_ID),
         N(NAS_PORT_TYPE)})

7) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(INVALID_CALLING_STATION_ID)})

8) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
```



```
         HDR(A,B), SK {})

9) A <-- B: EAP-Failure
```

Figure 13: EAP-IKEv2 with channel binding error (detected by EAP server)

Figure 14 shows an example of EAP-IKEv2 authentication sequence when the EAP peer detects an error in a channel binding procedure. The first two messages constitute the standard EAP identity exchange and are optional.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR(A,0), SAi1, KEi, Ni)

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
```

```
          HDR(A,B), SAr1, KEr, Nr, [CERTREQ,]
          N(CALLED_STATION_ID*))

   5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
          HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,], AUTH,
          N(CALLED_STATION_ID),
          N(CALLING_STATION_ID*),
          N(NAS_PORT_TYPE*)})

   6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
          HDR(A,B), SK {N(INVALID_CALLED_STATION_ID)})

   7) A <-- B: EAP-Failure
```

Figure 14: EAP-IKEv2 with channel binding error (detected by EAP peer)

Figure 15 shows an example of EAP-IKEv2 fast reconnection sequence with a successful channel binding procedure. The first two messages constitute the standard EAP identity exchange and are optional.

```
   1) A <-- B: EAP-Request/Identity

   2) A --> B: EAP-Response/Identity(Id)

   3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR, SK {SA, Ni, [KEi,]
          N(CALLING_STATION_ID*),
```



```
          N(NAS_PORT_TYPE*)})

   4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(HDR, SK {SA, Nr, [KEr,]
          N(CALLED_STATION_ID*),
          N(CALLING_STATION_ID),
          N(NAS_PORT_TYPE)})

   5) A <-- B: EAP-Response/EAP-Type=EAP-IKEv2(
          HDR(A,B), SK {N(CALLED_STATION_ID)})

   6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(HDR(A,B), SK {})

   7) A <-- B: EAP-Success
```

Figure 15: Fast reconnect with channel binding error (fast reconnect)

Figure 16 shows an example of EAP-IKEv2 fast reconnect sequence when the EAP server detects an error in a channel binding procedure. The first two messages constitute the standard EAP identity exchange and are optional. In this case, message 7) and 8) MUST constitute an INFORMATIONAL exchange.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR, SK {SA, Ni, [KEi,]
         N(CALLING_STATION_ID*),
         N(NAS_PORT_TYPE*)})

4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(HDR, SK {SA, Nr, [KEr,]
         N(CALLED_STATION_ID*),
         N(CALLING_STATION_ID),
         N(NAS_PORT_TYPE)})

5) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(INVALID_CALLING_STATION_ID)})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {})

7) A <-- B: EAP-Failure
```

Figure 16: Fast reconnect with channel binding error (detected by EAP server)

Figure 17 shows an example of EAP-IKEv2 fast reconnect sequence when the EAP peer detects an error in a channel binding procedure. The first two messages constitute the standard EAP identity exchange and are optional.

```
1) A <-- B: EAP-Request/Identity

2) A --> B: EAP-Response/Identity(Id)

3) A <-- B: EAP-Request/EAP-Type=EAP-IKEv2(HDR, SK {SA, Ni, [KEi,]
         N(CALLING_STATION_ID*),
         N(NAS_PORT_TYPE*)})
```

```
4) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(HDR, SK {SA, Nr, [KEr,]
         N(CALLED_STATION_ID*),
         N(CALLING_STATION_ID),
         N(NAS_PORT_TYPE)})

5) A <-- B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(CALLED_STATION_ID)})

6) A --> B: EAP-Response/EAP-Type=EAP-IKEv2(
         HDR(A,B), SK {N(INVALID_CALLED_STATION_ID)})

7) A <-- B: EAP-Failure
```

Figure 17: Fast reconnect with channel binding error (detected by EAP peer)

13. Security Considerations

13.1 General Considerations

The security of EAP-IKEv2 is intentionally based on IKEv2 [Kau04]. Therefore, the security claims of EAP-IKEv2 are derived mainly from the security offered by the supported features of IKEv2.

IKEv2 provides an improvement over IKEv1 [RFC2409] as described in Appendix A of [Kau04]. Important for this document are the reduced number of initial exchanges, decreased latency of the initial exchange, and some other fixes (e.g., hash problem). IKEv2 is a cryptographically sound protocol that has received a considerable amount of expert review and that benefits from a long practical experience with IKE.

In addition, IKEv2 provides authentication and key exchange capabilities which allow an entity to use symmetric as well as asymmetric authentication within a single protocol. Such

flexibility is considered important for an EAP method and is provided by EAP-IKEv2.

[Per03] provides a good tutorial for IKEv2 design decisions.

13.2 Security Claims

Authentication mechanism:
Mutual authentication is supported based on either pre-shared symmetric keys or public/private key pairs. Besides certificates, plain public keys can be used. It is possible to use different types of authentication for the different directions within one authentication exchange. An example is the server using certificate-based authentication and the client using pre-shared secrets.

EAP-IKEv2 changes the roles regarding password usage: The EAP server acts as initiator, the remote peer as responder. This results in an exchange which protects user authentication (based on a shared secret derived from a user password) to the network through an already network (initiator-) authenticated, secured IKEv2 SA (see e.g. message 6 of Figure 1). This prevents an attacker from launching password-guessing attacks on the peer-generated AUTH value.
Therefore, dictionary attacks are not applicable in the context of EAP-IKEv2 in the case the EAP peer uses a password-derived shared secret.

Man-in-the-middle attacks discovered in the context of tunneled authentication protocols (see [AN03] and [PL+03]) are not applicable to EAP-IKEv2 as the extended authentication feature of IKEv2 is not supported by EAP-IKEv2. Hence, the cryptographic binding claim is not applicable.

Ciphersuite negotiation is supported as specified in IKEv2 for IKE-SAs. The negotiation for IPsec (Child) SAs is not supported, as such SAs are not generated by EAP-IKEv2.

Protected result indication as described in section 7.16 of [RFC3748] is optionally provided by EAP-IKEv2. In message 5 of figure 1 (full successful authentication) the EAP server authenticates to the client. Message 6 authenticates the client to the server, and the client by authenticating the server and by sending message 6 expresses that it is willing to accept access. The client, however, does not get a protected result indication from the server in this case. An attacker could potentially forge an EAP success/failure message which could result in DoS to the

client. In some situations, synchronization may be achieved by lower layer indications.

Protected result indication is optionally provided as specified

in section 12.
If this mechanism is not used, the recommended behavior for the client is to assume the correct establishment of a new IKE-SA after sending message 6, independent of the receipt of an EAP success/failure. In case of unsuccessful authentication, the server would answer with a notification (which, in case of the fast reconnect exchange, would be protected by the old IKE-SA). In case of a lost message 6, the server would retransmit message 5, indicating the message loss to the client.
The client implementation can minimize potential DoS risks due to missing protected result indications by assuming the correct establishment of a new IKE-SA after not receiving one of the above messages within a certain time window after sending message 6. In the fast reconnect case, the client needs to hold both the old and the new IKE-SA in parallel during this time window.

Session independence is optionally provided if the fast reconnect exchange includes the KE payloads (new Diffie-Hellman) as described in section 11, Figure 9.

Security claims:

Ciphersuite negotiation:	Yes
Mutual authentication:	Yes
Integrity protection:	Yes
Replay protection:	Yes
Confidentiality:	Yes
Key derivation:	Yes
Key strength:	Variable
Dictionary attack prot.:	Yes
Fast reconnect:	Yes
Crypt. binding:	N/A
Protected result ind.:	yes
Session independence:	yes
Fragmentation:	Yes
Channel binding:	Yes

14. IANA Considerations

This section provides guidance to the IANA regarding registration of values related to the EAP-IKEv2 protocol, in accordance with [RFC2434].

The following terms are used here with the meanings defined in [RFC2434]: "name space" and "registration".

The following policies are used here with the meanings defined in [RFC2434]: "Expert Review" and "Specification Required".

This document introduces one new Internet Assigned Numbers Authority (IANA) consideration:

- It requires IANA to allocate an EAP-Request/Response Type for EAP-IKEv2.

<TBD: IANA considerations for channel binding notify payloads>

15. Normative References

[RFC3748] Aboba, Blunk, Carlson and Levkowetz: "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[Kau04] C. Kaufman: "Internet Key Exchange (IKEv2) Protocol", internet draft, Internet Engineering Task Force, September 2004. Work in progress.

[RFC2119] S. Bradner: "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Internet Engineering Task Force, March 1997.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

16. Informative References

[AN03] N. Asokan, V. Niemi, and K. Nyberg: "Man-in-the-middle in tunnelled authentication", In the Proceedings of the 11th International Workshop on Security Protocols, Cambridge, UK, April 2003. To be published in the Springer-Verlag LNCS series.

[PL+03] J. Puthenkulam, V. Lortz, A. Palekar, D. Simon, and B. Aboba, "The compound authentication binding problem," internet draft, Internet Engineering Task Force, October 2003. Expired.

[RFC2409] D. Harkins, D. Carrel: "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[Per03] R. Perlman: "Understanding IKEv2: Tutorial, and rationale for decisions", internet draft, Internet Engineering Task Force, 2003. Expired.

[AS+05] B. Aboba, D. Simon, J. Arkko, P. Eronen and H. Levkowetz: "Extensible Authentication Protocol (EAP) Key Management Framework", internet draft, Internet Engineering Task Force, April, 2005. Work in progress.

[PS+03] A. Palekar, D. Simon, G. Zorn, H. Zhou and S. Josefsson: "Protected EAP Protocol (PEAP)", internet draft, Internet Engineering Task Force, July 2004. Work in progress.

Acknowledgments

We would like to thank Bernard Aboba, Jari Arkko, Guenther Horn, Paoulo Pagliusi and John Vollbrecht for their comments to this draft.

Additionally we would like to thank members of the PANA design team (namely D. Forsberg and A. Yegin) for their comments and input to the initial version of the draft.

Finally we would like to thank the members of the EAP keying design team for their discussion in the area of the EAP Key Management Framework.

Author's Addresses

Hannes Tschofenig
Siemens AG
Otto-Hahn-Ring 6
81739 Munich
Germany
EMail: Hannes.Tschofenig@siemens.com

Dirk Kroeselberg
Siemens AG
Otto-Hahn-Ring 6
81739 Munich
Germany
EMail: Dirk.Kroeselberg@siemens.com

Yoshihiro Ohba
Toshiba America Research, Inc.

1 Telcordia Drive
Piscataway, NJ 08854

USA

Phone: +1 732 699 5305
EMail: yohba@tari.toshiba.com

Florent Bersani
France Telecom R&D
38, rue du General Leclerc
Issy-Les-Moulineaux 92794 Cedex 9
FR

EMail: florent.bersani@francetelecom.com

Intellectual Property Statement

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND

THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Acknowledgment

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