Network Working Group
Request for Comments: 5106
Category: Experimental

H. Tschofenig
D. Kroeselberg
Nokia Siemens Networks
A. Pashalidis
NEC
Y. Ohba
Toshiba
F. Bersani
France Telecom

February 2008

The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Abstract

This document specifies EAP-IKEv2, an Extensible Authentication Protocol (EAP) method that is based on the Internet Key Exchange (IKEv2) protocol. EAP-IKEv2 provides mutual authentication and session key establishment between an EAP peer and an EAP server. It supports authentication techniques that are based on passwords, high-entropy shared keys, and public key certificates. EAP-IKEv2 further provides support for cryptographic ciphersuite negotiation, hash function agility, identity confidentiality (in certain modes of operation), fragmentation, and an optional "fast reconnect" mode.

Table of Contents

1. Introduction 3
2. Terminology 4
3. Protocol Overview 6
4. Fast Reconnect 9
5. Key Derivation 12
6. Session ID, Peer ID, and Server ID 13
7. Error Handling 13
8. Specification of Protocol Fields 16
8.1. The Flags, Message Length, and Integrity Checksum Data Fields 17
8.2. EAP-IKEv2 Header 19
8.3. Security Association Payload 19
8.4. Key Exchange Payload 20
8.5. Nonce Payload 20
8.6. Identification Payload 20
8.7. Certificate Payload 20
8.8. Certificate Request Payload 20
8.9. Encrypted Payload 20
8.10. Authentication Payload 20
8.11. Notify Payload 21
8.12. Next Fast-ID Payload 21
9. Payload Types and Extensibility 22
10. Security Considerations 22
10.1. Protected Ciphersuite Negotiation 23
10.2. Mutual Authentication 23
10.3. Integrity Protection 23
10.4. Replay Protection 23
10.5. Confidentiality 23
10.6. Key Strength 24
10.7. Dictionary Attack Resistance 24
10.8. Fast Reconnect 25
10.9. Cryptographic Binding 25
10.10. Session Independence 25
10.11. Fragmentation 26
10.12. Channel Binding 26
10.13. Summary 26
11. IANA Considerations 27
12. Contributors 28
13. Acknowledgements 28
14. References 29
14.1. Normative References 29
14.2. Informative References 29
Appendix A 30

1. Introduction

This document specifies EAP-IKEv2, an EAP method that is based on the Internet Key Exchange Protocol version 2 (IKEv2) [1]. EAP-IKEv2 provides mutual authentication and session key establishment between an EAP peer and an EAP server. It supports authentication techniques that are based on the following types of credentials:

- Asymmetric key pairs: these are public/private key pairs where the public key is embedded into a digital certificate, and the corresponding private key is known only to a single party.

- Passwords: these are low-entropy bit strings that are known to both the server and the peer.

- Symmetric keys: these are high-entropy bit strings that are known to both the server and the peer.

It is possible to use a different authentication credential (and thereby technique) for each direction, e.g., the EAP server may authenticate itself using a public/private key pair and the EAP client may authenticate itself using a symmetric key. In particular, the following combinations are expected to be used in practice; these are referred to as "use cases" or "modes" in the remainder of this document:

1. EAP server: asymmetric key pair, EAP peer: asymmetric key pair

2. EAP server: asymmetric key pair, EAP peer: symmetric key

3. EAP server: asymmetric key pair, EAP peer: password

4. EAP server: symmetric key, EAP peer: symmetric key

Note that in use cases 2 and 4, a symmetric key is assumed to be chosen uniformly at random from its key space; it is therefore assumed that symmetric keys are not derived from passwords. Deriving a symmetric key from a password is insecure when used with mode 4 since the exchange is vulnerable to dictionary attacks, as described in more detail in Section 10.7. Also note that in use case 3, the EAP server must either have access to all passwords in plaintext, or, alternatively, for each password store, the value prf(password,"Key Pad for EAP-IKEv2") for all supported pseudorandom functions (also

see Section 8.10 below and Section 2.15 of [1]). Other conceivable use cases are not expected to be used in practice due to key management issues, and have not been considered in this document.

Note that the IKEv2 protocol is able to carry EAP exchanges. By contrast, EAP-IKEv2 does not inherit this capability. That is, it is not possible to tunnel EAP methods inside EAP-IKEv2. Also note that the set of functionality provided by EAP-IKEv2 is similar, but not identical, to that provided by other EAP methods such as, for example, EAP-TLS [6].

The remainder of this document is structured as follows:

- Section 2 provides an overview of the terminology and the abbreviations used in this document.
- Section 3 provides an overview of the full EAP-IKEv2 exchange and thereby specifies the protocol message composition.
- Section 4 specifies the optional EAP-IKEv2 "fast reconnect" mode of operation.
- Section 5 specifies how exportable session keys are derived.
- Section 6 specifies how the Session-ID, Peer-ID, and Server-ID elements are derived.
- Section 7 specifies how errors that may potentially occur during protocol execution are handled.
- Section 8 specifies the format of the EAP-IKEv2 data fields. Section 8.1 describes how fragmentation is handled in EAP-IKEv2.
- Section 9 specifies the payload type values and describes protocol extensibility.
- Section 10 provides a list of claimed security properties.

2. Terminology

This document makes use of terms defined in [2] and [1]. In addition, 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 [3].

A list of abbreviations that are used in this document follows.

AUTH:

Denotes a data field containing either a Message Authentication Code (MAC) or a signature. This field is embedded into an Authentication payload, defined in Section 8.10.

CERT:

Public key certificate or similar structure.

CERTREQ:

Certificate Request.

NFID:

Next Fast-ID payload (see Sections 4 and 8.12)

EMSK:

Extended Master Session Key, defined in [2].

HDR:

EAP-IKEv2 header, defined in Section 8.2.

I:

Initiator, the party that sends the first message of an EAP-IKEv2 protocol run. This is always the EAP server.

MAC:

Message Authentication Code. The result of a cryptographic operation that involves a symmetric key.

MSK:

Master Session Key, defined in [2].

prf:

Pseudorandom function: a cryptographic function whose output is assumed to be indistinguishable from that of a truly random function.

R:

Responder, the party that sends the second message of an EAP-IKEv2 protocol run. This is always the EAP peer.

SA:

Security Association. In this document, SA denotes a type of payload that is used for the negotiation of the cryptographic algorithms that are to be used within an EAP-IKEv2 protocol run. Specifically, SAi denotes a set of choices that are accepted by an initiator, and SAr denotes the choice of the responder.

Signature:

The result of a cryptographic operation that involves an asymmetric key. In particular, it involves the private part of a public/private key pair.

SK:

Session Key. In this document, the notation SK{x} denotes that x is embedded within an Encrypted payload, i.e., that x is encrypted and integrity-protected using EAP-IKEv2 internal keys. These keys are different in each direction.

SK_xx:

EAP-IKEv2 internal key, defined in Section 2.14 of [1].

SKEYSEED:

Keying material, defined in Section 2.14 of [1].

3. Protocol Overview

In this section, the full EAP-IKEv2 protocol run is specified. All messages are sent between two parties, namely an EAP peer and an EAP server. In EAP-IKEv2, the EAP server always assumes the role of the initiator (I), and the EAP peer that of the responder (R) of an exchange.

The semantics and formats of EAP-IKEv2 messages are similar, albeit not identical, to those specified in IKEv2 [1] for the establishment of an IKE Security Association. The full EAP-IKEv2 protocol run consists of two roundtrips that are followed by either an EAP-Success or an EAP-Failure message. An optional roundtrip for exchanging EAP identities may precede the two exchanges.

1. R<-I: EAP-Request/Identity

2. R->I: EAP-Response/Identity(Id)

3. R<-I: EAP-Req (HDR, SAi, KEi, Ni)

4. R->I: EAP-Res (HDR, SAr, KEr, Nr, [CERTREQ], [SK{IDr}])

5. R<-I: EAP-Req (HDR, SK{IDi, [CERT], [CERTREQ], [NFID], AUTH})

6. R->I: EAP-Res (HDR, SK{IDr, [CERT], AUTH})

7. R<-I: EAP-Success

Figure 1: EAP-IKEv2 Full, Successful Protocol Run

Figure 1 shows the full EAP-IKEv2 protocol run, including the optional EAP identity exchange (messages 1 and 2). A detailed specification of the message composition follows.

Messages 1 and 2 are a standard EAP Identity Request and Response, as defined in [2]. Message 3 is the first EAP-IKEv2-specific message. With this, the server starts the actual EAP authentication exchange. It contains the initiator Security Parameter Index (SPI) in the EAP-IKEv2 header (HDR) (the initiator selects a new SPI for each protocol run), the set of cryptographic algorithms the server is willing to accept for the protection of EAP-IKEv2 traffic (encryption and integrity protection), and the derivation of the session key. This set is encoded in the Security Association payload (SAi). It also contains a Diffie-Hellman payload (KEi), and a Nonce payload (Ni).

When the peer receives message 3, it selects a set of cryptographic algorithms from the ones that are proposed in the message. In this overview, it is assumed that an acceptable such set exists (and is thus selected), and that the Diffie-Hellman value KEi belongs to an acceptable group. The peer then generates a non-zero Responder SPI value for this protocol run, its own Diffie-Hellman value (KEr) and nonce (Nr), and calculates the keys SKEYSEED, SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, and SK_pr, according to Section 2.14 of [1]. The peer then constructs message 4. In the context of use cases 1, 2, and 3, the peer’s local policy MAY dictate the inclusion of the optional CERTREQ payload in that message, which gives a hint to the server to include a certificate for its public key in its next message. In the context of use case 4, the peer MUST include the optional SK{IDr} payload, which contains its EAP-IKEv2 identifier, encrypted and integrity-protected within an Encrypted payload. The keys used to construct this Encrypted payload are SK_er (for encryption) and SK_ar (for integrity protection), in accordance with

[1]. The responder's EAP-IKEv2 identifier (IDr) is likely to be needed in these use cases by the server in order to select the correct symmetric key or password for the construction of the AUTH payload of message 5.

Upon reception of message 4, the server also computes SKEYSEED, SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, and SK_pr, according to Section 2.14 of [1]. If an SK{IDr} payload is included, the server decrypts it and verifies its integrity with the corresponding keys. In this overview, decryption and verification is assumed to succeed. The server then constructs message 5, which contains only the EAP-IKEv2 header followed by a single Encrypted payload. The keys used to generate the encrypted payload MUST be SK_ei (for encryption) and SK_ai (for integrity protection), in accordance with [1]. The initiator MUST embed at least two payloads in the Encrypted Payload, as follows. An Identification payload with the initiator's EAP-IKEv2 identifier MUST be embedded in the Encrypted payload. The Authentication payload MUST be embedded in the Encrypted payload. A Certificate payload, and/or a Certificate Request payload, MAY also be embedded in the Encrypted payload. Moreover, a Next Fast-Reconnect Identifier payload MAY also be embedded in the Encrypted payload. Message 5 is sent to the responder.

Upon reception of message 5, the responder (EAP peer) authenticates the initiator (EAP server). The checks that are performed to this end depend on the use case, local policies, and are specified in [1]. These checks include (but may not be limited to) decrypting the Encrypted payload, verifying its integrity, and checking that the Authentication payload contains the expected value. If all checks succeed (which is assumed in this overview), then the responder constructs message 6. That message MUST contain the EAP-IKEv2 header followed by a single Encrypted payload, in which at least two further payloads MUST be embedded, as shown in Figure 1.

Upon reception of message 6, the initiator (EAP server) authenticates the responder (EAP peer). As above, the checks that are performed to this end depend on the use case, local policies, and MUST include decryption and verification of the Encrypted payload, as well as checking that the Authentication payload contains the expected value. If the optional SK{IDr} payload was included in message 4, the EAP server MUST also ensure that the IDr payload in message 6 is identical to that in message 4.

If authentication succeeds, an EAP-Success message is sent to the responder as message 7. The EAP server and the EAP peer generate a Master Session Key (MSK) and an Extended Master Session Key (EMSK) after a successful EAP-IKEv2 protocol run, according to Section 5.

4. Fast Reconnect

This section specifies a "fast reconnect" mode of operation for EAP-IKEv2. This mode is mandatory to implement, but optional to use. The purpose of fast reconnect is to enable an efficient re-authentication procedure that also results in a fresh MSK and EMSK. The "fast reconnect" mode can only be used where an EAP-IKEv2 security context already exists at both the server and the peer, and its usage is subject to the local policies. In other words, it can only be used by an EAP server/EAP peer pair that has already performed mutual authentication in a previous EAP-IKEv2 protocol run.

The fast reconnect mode makes use of dedicated "fast reconnect EAP identifiers". The idea is that the server indicates its willingness to engage in "fast reconnect" protocol runs in the future by including the optional "Next Fast-ID" (NFID) payload in message 5 of a "full" protocol run (see Figure 1), or in message 3 of a "fast reconnect" protocol run (see Figure 2). This NFID payload contains a special EAP identity, denoted Fast Reconnect Identity (FRID) as the Network Access Identifier (NAI) in the EAP-Response/Identity message. The FRID contains an obfuscated username part and a realm part. When generating a FRID, the following aspects should be considered:

The FRID and therefore the pseudonym usernames are generated by the EAP server. The EAP server produces pseudonym usernames in an implementation-dependent manner. Only the EAP server needs to be able to map the pseudonym username to the permanent identity.

EAP-IKEv2 includes no provisions to ensure that the same EAP server that generated a pseudonym username will be used on the authentication exchange when the pseudonym username is used. It is recommended that the EAP servers implement some centralized mechanism to allow all EAP servers of the home operator to map pseudonyms generated by other severs to the permanent identity. If no such mechanism is available, then the EAP server, failing to understand a pseudonym issued by another server, can request the peer to send the permanent identity.

When generating FRIDs, the server SHOULD choose a fresh and unique FRID that is different from the previous ones that were used after the same full authentication exchange. The FRID SHOULD include a random component in the username part. The random component works as a reference to the security context. Regardless of the construction method, the pseudonym username MUST conform to the grammar specified for the username portion of an NAI. Also, the FRID MUST conform to the NAI grammar [4]. The EAP servers, which subscribers of an operator can use, MUST ensure that the username part of a FRIDs that they generate are unique.

The peer MAY use the FRID to indicate to start a "fast reconnect" protocol run. The EAP Identity Response MUST be sent at the beginning of a "fast reconnect" protocol run. If, in the previous successful "full" (resp. "fast reconnect") EAP-IKEv2 protocol execution, the server had not included an NFID payload in message 5 (resp. 3), then the peer MUST NOT start a fast reconnect protocol run. On reception of FRID, the server maps it to an existing EAP-IKEv2 security context. Depending on local policy, the server either proceeds with the "fast reconnect" protocol run, or proceeds with message 3 of a "full" protocol run. If the server had advertised the FRID in the previous EAP-IKEv2 protocol execution, it SHOULD proceed with a "fast reconnect" protocol run. The peer MUST be able to correctly handle a message 3 of a "full" protocol run, even if it indicated a FRID in its EAP Identity Response.

Because the peer may fail to save a FRID that was sent in the NFID payload (for example, due to malfunction), the EAP server SHOULD maintain, at least, the most recently used FRID in addition to the most recently issued FRID. If the authentication exchange is not completed successfully, then the server MUST NOT overwrite the FRID that was issued during the most recent successful authentication exchange.

The EAP-IKEv2 fast reconnect exchange is similar to the IKE-SA rekeying procedure, as specified in Section 2.18 of [1]. Thus, it uses a CREATE_CHILD_SA request and response. The SPIs on those two messages would be the SPIs negotiated on the previous exchange. During fast reconnect, the server and the peer MAY exchange fresh Diffie-Hellman values.

1. R<-I: EAP-Request/Identity

2. R->I: EAP-Response/Identity(FRID)

3. R<-I: EAP-Req(HDR, SK{SA, Ni, [KEi], [NFID]})

4. R->I: EAP-Res(HDR, SK{SA, Nr, [KEr]})

5. R<-I: EAP-Success

Figure 2: Fast Reconnect Protocol Run

Figure 2 shows the message exchange for the EAP-IKEv2 fast reconnect mode. As in the full mode, the EAP server is the initiator and the EAP peer is the responder. The first two messages constitute the standard EAP identity exchange. Note that, in order to use the "fast reconnect" mode, message 2 MUST be sent. This is in order to enable the peer to indicate its "fast reconnect" identity FRID in message 2.

If the server can map the FRID to an existing EAP-IKEv2 context it proceeds with message 3. Note that, in this message, the server MAY embed an NFID payload into the encrypted payload to provide a new FRID to the peer. The server MAY choose to perform a full EAP-IKEv2 run, in which case, it would respond with a message that conforms to the format of message 3 in Figure 1.

Messages 3 and 4 establish a new EAP-IKEv2 security context. In message 3, the initiator MUST select a new (non-zero) value for the SPI field in each proposal substructure in the SA payload (see Section 3.3 of [1]). The value of the IKE_SA Responder's SPI field in HDR MUST be the one from the previous successful EAP-IKEv2 protocol run. The nonce inside the Nonce payload (Ni) MUST be fresh, and the Diffie-Hellman value inside the Diffie-Hellman payload (if present, KEi) MUST also be fresh. If present, the Diffie-Hellman value MUST be drawn from the same group as the Diffie-Hellman value in the previous successful full EAP-IKEv2 protocol run. Note that the algorithms and keys that are used to construct the Encrypted payload in message 3 are the same as in the previous successful EAP-IKEv2 protocol run.

Upon reception of message 3, the responder (EAP peer) decrypts and verifies the Encrypted payload. If successful (as assumed in Figure 2), it constructs message 4 in a fashion similar to the construction of message 3. The responder MUST choose a new (non-zero) value for the SPI field in each proposal substructure. Upon reception of message 4, the initiator (EAP server) decrypts and verifies the Encrypted payload. If a correct message 4 is received, then this protocol run is deemed successful, and the server responds with an EAP-Success message (message 5).

After successful EAP-IKEv2 fast reconnect protocol run, both the initiator and the responder generate fresh keying material that is used for the protection of subsequent EAP-IKEv2 traffic. Furthermore, both the initiator and the responder MUST generate a fresh MSK and EMSK and export them.

The new EAP-IKEv2-specific keying material is computed in the same way as in the full EAP-IKEv2 protocol run, and in accordance with Section 2.18 of [1]. That is, SKEYSEED is computed as SKEYSEED = prf(SK_d (old), [g^ir (new)] | Ni | Nr), where SK_d (old) is the key SK_d from the previous successful EAP-IKEv2 protocol run, Ni and Nr are the nonces (without the Nonce payload headers) that were exchanged in messages 3 and 4, and g^ir (new) is the newly computed Diffie-Hellman key, if both the values KEi and KEr were present in messages 3 and 4. The remaining EAP-IKEv2-specific keys (SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, and SK_pr) are generated as in the full EAP-IKEv2 protocol run.

The generation of a fresh MSK and EMSK follows the generation of the EAP-IKEv2-specific keys and adheres to the rules in Section 5.

Note 1: In EAP-IKEv2, the EAP server initiates the fast reconnect mode and thereby causes fresh session keys to be established.

Note 2: It is conceivable that an adversary tries to launch a replay attack against the EAP-IKEv2 fast reconnect mode of operation. In particular, the adversary may try to send a previously captured message 3 in a subsequent fast reconnect protocol run. This replay attempt will, however, fail because the keys that the responder will use to verify and decrypt the Encrypted payload are changed with every successful reconnect protocol run.

5. Key Derivation

This section describes how the Master Session Key (MSK) and the Extended Master Session Key (EMSK) are derived in EAP-IKEv2. It is expected that the MSK and the EMSK are exported by the EAP-IKEv2 process and be used in accordance with the EAP keying framework [7].

During an EAP-IKEv2 protocol run, the initiator and the responder generate a number of keys, as described above and in accordance with Section 2.14 of [1]. The generation of these keys is based on a pseudorandom function (prf) that both parties have agreed to use and that is applied in an iterative fashion. This iterative fashion is specified in Section 2.13 of [1] and is denoted by prf+.

In particular, following a successful EAP-IKEv2 protocol run, both parties generate 128 octets of keying material, denoted KEYMAT, as KEYMAT = prf+(SK_d, Ni | Nr), where Ni and Nr are the nonces (just payload without headers) from messages 3 and 4 shown in Figure 1 (in the context of a full EAP-IKEv2 protocol run) or Figure 2 (in the context of a fast reconnect EAP-IKEv2 protocol run). Note that only the nonces are used, i.e., not the entire Nonce payload that contains them.

The first 64 octets of KEYMAT are exported as the EAP MSK, and the second 64 octets are exported as the EMSK.

The MSK and EMSK MUST NOT be generated unless an EAP-IKEv2 protocol run completes successfully. Note that the EAP-IKEv2 method does not produce an initialisation vector [7].

6. Session ID, Peer ID, and Server ID

The EAP key management framework [7] requires that EAP methods export three information elements, called the Session-ID, the Peer-ID, and the Server-ID. In EAP-IKEv2, these elements are derived as follows:

- The Session-ID is constructed and exported as the concatenation of the following three elements, in this order: (a) the EAP Code Type for EAP-IKEv2 (to be defined by IANA), (b) the contents of the Nonce Data field of the Nonce Payload Ni from message 3, (c) the contents of the Nonce Data field of the Nonce Payload Nr from message 4.

- In case of a full EAP-IKEv2 protocol run, the Peer-ID is constructed and exported as the content of the Identification Data field of the Identification Payload IDr from message 6. Note that only the "actual" identification data is exported, as indicated in the Payload Length field; if the Identification Data field contains any padding, this padding is ignored. In case of a "fast reconnect" protocol run, the Peer-ID field is constructed in exactly the same manner, where message 6 refers to the full EAP-IKEv2 protocol run that originally established the security context between the EAP peer and EAP server.

- In case of a full EAP-IKEv2 protocol run, the Server-ID is constructed and exported as the contents of the Identification Data field of the Identification Payload IDi from message 5. Note that only the "actual" identification data is exported, as indicated in the Payload Length field; if the Identification Data field contains any padding, this padding is ignored. In case of a "fast reconnect" protocol run, the Server-ID field is constructed in exactly the same manner, where message 5 refers to the full EAP-IKEv2 protocol run that originally established the security context between the EAP peer and EAP server.

7. Error Handling

This section specifies how errors are handled within EAP-IKEv2. For conveying error information from one party to the other, the Notify payload is defined and used (see Section 8.11).

If, in a full EAP-IKEv2 protocol run, authentication fails (i.e., the verification of the AUTH field fails at the server or the peer), but no other errors have occurred, the message flow deviates from that described in Section 3. The message flows in the presence of authentication failures are specified in Appendix A.

If, in message 3 of a full EAP-IKEv2 protocol run (see Figure 1), the responder receives a Diffie-Hellman value (KEi) that belongs to a group that is not supported (and in the absence of other errors), then the responder MUST send a message of the form shown in Figure 3 to the initiator. This effectively becomes message 4 in the full protocol run.

1. R<-I: EAP-Request/Identity

2. R->I: EAP-Response/Identity(Id)

3. R<-I: EAP-Req (HDR, SAi, KEi, Ni)

4. R->I: EAP-Res (HDR, N(INVALID_KE_PAYLOAD))

Figure 3: Error Handling in Case of Unsupported D-H Value

The above message consists of the EAP-IKEv2 header and a Notification payload with the value of the Notify Message Type field value set to 17 (INVALID_KE_PAYLOAD). There is a two-octet value associated with this notification: the number of the selected DH Group in big endian order, as specified in Section 3.10.1 of [1]. This number MUST represent a DH group that is supported by both the initiator and the responder.

If, during a full EAP-IKEv2 protocol run (see Figure 1), the initiator receives a message conforming to Figure 3 instead of the usual message 4, then it MUST check whether or not the indicated DH group was proposed in message 3. If it was not, then the initiator MUST silently discard the message. Otherwise, the protocol continues with a new message 3 that the initiator sends to the peer. In this new message 3, the initiator MUST use a Diffie-Hellman value that is drawn from the group that is indicated in the Notify payload of message 4 in Figure 3.

If, in the context of use case 4 and during a full EAP-IKEv2 protocol run (see Figure 1), the initiator receives, in message 4, an SK{IDr} payload that decrypts to a non-existent or unauthorised EAP-IKEv2 responder identifier IDr*, then the server SHOULD continue the protocol with a message conforming to the format of message 5. The AUTH payload in that message SHOULD contain a value that is computationally indistinguishable from a value that it would contain if IDr* was valid and authorised. This can be accomplished, for example, by generating a random key and calculating AUTH as usual (however, this document does not mandate a specific mechanism). Only after receiving message 6, the server SHOULD respond with an

authentication failure notification, i.e., a message conforming to message 6 in Figure 10. The purpose of this behaviour is to prevent an adversary from probing the EAP-IKEv2 peer identifier space.

If, in the context of use cases 1, 2, or 3 and during a full EAP-IKEv2 protocol run (see Figure 1), the initiator receives, in message 4, an SK{IDr} payload that decrypts to an EAP-IKEv2 responder identifier IDr*, then the server MUST continue the protocol as usual (note that such a payload would not be required in these use cases). The server MUST compare IDr* with the IDr received in message 6 and, in case of a mismatch, MUST respond with an authentication failure notification, i.e., a message conforming to message 6 in Figure 10. If no mismatch is detected, normal processing applies.

Other errors do not trigger messages with Notification payloads to be sent, and MUST be treated as if nothing happened (i.e., the erroneous EAP-IKEv2 packet MUST be silently discarded). This includes situations where at least one of the following conditions is met, with respect to an incoming EAP-IKEv2 packet.

- The packet contains an Encrypted payload that, when decrypted with the appropriate key, yields an invalid decryption.
- The packet contains an Encrypted payload with a Checksum field that does not verify with the appropriate key.
- The packet contains an Integrity Checksum Data field (see *Figure 4) that is incorrect.
- The packet does not contain a compulsory field.
- A field in the packet contains an invalid value (e.g., an invalid combination of flags, a length field that is inconsistent with the real length of the field or packet, or the responder’s choice of a cryptographic algorithm is different to NONE and any of those that were offered by the initiator).
- The packet contains an invalid combination of fields (e.g., it contains two or more Notify payloads with the same Notify Message Type value, or two or more Transform substructures with the same Transform Type and Transform ID value).
- The packet causes a defragmentation error.
- The format of the packet is invalid.

- o The identity provided by the EAP peer in the EAP-Response/Identity cannot be associated with either an established security context (in case of a fast reconnect) or with a real user account (in case of a full protocol exchange). In that case, the packet is silently discarded. With an outstanding message from the EAP server, the client may either retransmit the previous request or, in case of a fast reconnect, assume that state information was deleted (e.g., due to garbage collection) at the EAP server and fall back to a previously used FRID or to the full protocol exchange.

If an incoming packet contains an error for which a behaviour is specified in this section, and an error that, in the absence of the former error, would cause the packet to be silently discarded, then the packet MUST be silently discarded.

8. Specification of Protocol Fields

In this section, the format of the EAP-IKEv2 data fields and applicable processing rules are specified. Figure 4 shows the general packet format of EAP-IKEv2 messages, and the embedding of EAP-IKEv2 into EAP. The EAP-IKEv2 messages are embedded in the Data field of the standard EAP Request/Response packets. The Code, Identifier, Length, and Type fields are described in [2]. The EAP Type for this EAP method is 49.

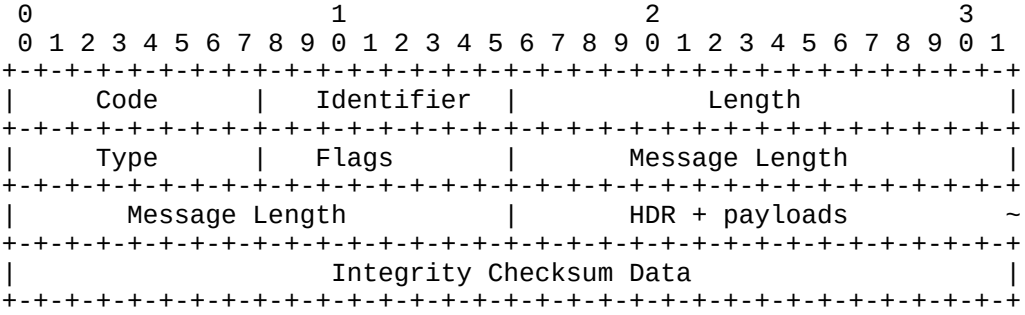

Figure 4: General Packet Format of EAP-IKEv2

The Flags field is always present and is used for fragmentation support, as described in Section 8.1. The Message Length field is not always present; its presence is determined by a certain flag in the Flags field, as described in Section 8.1. The field denoted as "HDR + payloads" in Figure 4 contains the EAP-IKEv2 header (see Section 8.2), followed by the number of payloads, in accordance with the composition of EAP-IKEv2 messages, as described in the previous

sections. Note that each payload begins with a generic payload header that is specified in Section 3.2 of [1].

The Integrity Checksum Data field is not always present; its presence is determined by a certain flag in the Flags field, as described in Section 8.1.

In the remainder of this section, the protocol fields that are used in EAP-IKEv2 are specified. This specification heavily relies on the IKEv2 specification [1], and many fields are constructed, formatted, and processed in way that is almost identical to that in IKEv2. However, certain deviations from standard IKEv2 formatting and processing exist. These deviations are highlighted in the remainder of this section.

8.1. The Flags, Message Length, and Integrity Checksum Data Fields

This section describes EAP-IKEv2 fragmentation, and specifies the encoding and processing rules for the Flags, Message Length, and Integrity Checksum Data field shown in Figure 4.

Fragmentation support in EAP-IKEv2 is provided by the Flags and Message Length fields shown in Figure 4. These are encoded and used as follows:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|L M I 0 0 0 0 0|
+-+-+-+-+-+-+-+-+

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

Figure 5: Flags Field

The Flags field is defined in Figure 5. Only the first three bits (0-2) are used; all remaining bits MUST be set to zero and ignored on receipt. The L flag indicates the presence of a Message Length field, and the M flag indicates whether or not the current EAP message has more fragments. In particular, if the L bit is set, then a Message Length field MUST be present in the EAP message, as shown in Figure 4. The Message Length field is four octets long and contains the length of the entire message (i.e., the length of the EAP Data field.). Note that, in contrast, the Length field shown in Figure 4 contains the length of only the current fragment. (Note that there exist two fields that are related to length: the Length

field, which is a generic EAP field, and the Message Length field, which is an EAP-IKEv2-specific field.) If the L bit is not set, then the Message Length field MUST NOT be present.

The M flag MUST be set on all fragments except the last one. In the last fragment, the M flag MUST NOT be set. Reliable fragment delivery is provided by the retransmission mechanism of EAP as described below.

When an EAP-IKEv2 peer receives an EAP-Request packet with the M bit set, it MUST respond with an EAP-Response with EAP-Type=EAP-IKEv2 and no data. This serves as a fragment ACK. The EAP server MUST wait until it receives the EAP-Response before sending another fragment. In order to prevent errors in processing of fragments, the EAP server MUST increment the Identifier field for each fragment contained within an EAP-Request, and the peer MUST include this Identifier value in the fragment ACK contained within the EAP-Response. Retransmitted fragments will contain the same Identifier value.

Similarly, when the EAP server receives an EAP-Response with the M bit set, it MUST respond with an EAP-Request with EAP-Type=EAP-IKEv2 and no data. This serves as a fragment ACK. The EAP peer MUST wait until it receives the EAP-Request before sending another fragment. In order to prevent errors in the processing of fragments, the EAP server MUST increment the Identifier value for each fragment ACK contained within an EAP-Request, and the peer MUST include this Identifier value in the subsequent fragment contained within an EAP-Response.

The Integrity Checksum Data field contains a cryptographic checksum that covers the entire EAP message, starting with the Code field, and ending at the end of the EAP Data field. This field, shown in Figure 4, is present only if the I bit is set in the Flags field. The Integrity Checksum Data field immediately follows the EAP Data field without padding.

Whenever possible, the Integrity Checksum Data field MUST be present (and the I bit set) for each fragment, including the case where the entire EAP-IKEv2 message is carried in a single fragment. The algorithm and keys that are used to compute the Integrity Checksum Data field MUST be identical to those used to compute the Integrity Checksum Data field of the Encrypted Payload (see Section 8.9). That is, the algorithm and keys that were negotiated and established during this EAP-IKEv2 protocol run are used. Note that this means that different keys are used to compute the Integrity Checksum Data field in each direction. Also note that, for messages where this

algorithm and the keys are not yet established, the Integrity Checksum Data field cannot be computed and is therefore not included. This applies, for example, to messages 3 and 4 in Figure 1.

In order to minimize the exposure to denial-of-service attacks on fragmented packets, messages that are not protected with an Integrity Checksum Data field SHOULD NOT be fragmented. Note, however, that those packets are not likely to be fragmented anyway since they do not carry certificates.

8.2. EAP-IKEv2 Header

The EAP-IKEv2 header, denoted HDR in this specification, is constructed and formatted according to the rules specified in Section 3.1 of [1].

In the first EAP-IKEv2 message that is sent by the initiator (message 3 in Figure 1), the IKE_SA Responder's SPI field is set to zero. This is because, at this point in time, the initiator does not know what SPI value the responder will choose for this protocol run. In all other messages, both SPI fields MUST contain non-zero values that reflect the initiator- and responder-chosen SPI values.

In accordance with [1], for this version of EAP-IKEv2, the MjVer (major version) and MnVer (minor version) fields in the header MUST be 2 and 0 respectively. The value of the Exchange Type field MUST be set to 34 (IKE_SA_INIT) in messages 3 and 4, and to 35 (IKE_SA_AUTH) in messages 5 and 6 in Figure 1. In messages 3 and 4 in Figure 2, this value MUST be set to 36 (CREATE_CHILD_SA).

The Flags field of the EAP-IKEv2 header is also constructed according to Section 3.1 of [1]. Note that this is not the same field as the Flags field shown in Figure 4.

The Message ID field is constructed as follows. Messages 3 and 4 in a full protocol run MUST carry Message ID value 0. Messages 5 and 6 in a full protocol run (see Figure 1) MUST carry Message ID value 1. Messages 3 and 4 in a fast reconnect protocol run MUST carry Message ID value 2.

8.3. Security Association Payload

The SA payload is used for the negotiation of cryptographic algorithms between the initiator and the responder. The rules for its construction adhere to [1]; in particular, Sections 2.7 and 3.3.

In EAP-IKEv2, all Proposal Substructures in the SA payload MUST carry Protocol ID value 1 (IKE).

8.4. Key Exchange Payload

The Key Exchange payload, denoted KEi if constructed by the initiator and KEr if constructed by the responder, is formatted according to the rules specified in Section 3.4 of [1].

8.5. Nonce Payload

The Nonce payload, denoted Ni if constructed by the initiator and Nr if constructed by the responder, is constructed and formatted according to the rules specified in Section 3.9 of [1].

8.6. Identification Payload

The Identification payload, denoted IDi if it contains an identifier for the initiator and IDr if it contains an identifier for the responder, is constructed and formatted according to the rules specified in Section 3.5 of [1].

8.7. Certificate Payload

The Certificate payload, denoted CERT, is constructed and formatted according to the rules specified in Section 3.6 of [1]. Note that certain certificate encodings for the EAP server certificate, e.g., those that need to be resolved via another network protocol, cannot be used in some typical EAP-IKEv2 deployment scenarios. A user, for example, that authenticates himself by means of EAP-IKEv2 in order to obtain network access, cannot resolve the server certificate at the time of EAP-IKEv2 protocol execution.

8.8. Certificate Request Payload

The Certificate Request payload, denoted CERTREQ, is constructed and formatted according to the rules specified in Section 3.7 of [1].

8.9. Encrypted Payload

The Encrypted payload, denoted SK{...}, is constructed and formatted according to the rules specified in Section 3.14 of [1].

8.10. Authentication Payload

The Authentication payload, denoted AUTH, is constructed and formatted according to the rules specified in Sections 2.15 and 3.8 of [1].

The contents of the Authentication payload depend on which party generates this field, the use case, and the algorithm that

corresponds to the credential (asymmetric key, symmetric key, or password) that this party uses to authenticate itself. The Authentication payload contains either a MAC or a signature.

If the party that generates the Authentication payload authenticates itself based on a shared secret (i.e., a password or a symmetric key), then the Authentication payload MUST contain a MAC. This MAC is calculated using a key that is derived from the shared secret, according to Section 2.15 of [1]. According to that section, the shared secret is padded with the string "Key Pad for IKEv2" as part of this key derivation. For the EAP-IKEv2 method, this rule is overridden, in that the padding string is redefined as "Key Pad for EAP-IKEv2". The latter padding string MUST be used for the derivation of the MAC key from a shared secret in the context of EAP-IKEv2. This is done in order to avoid the same MAC key to be used for both IKEv2 and EAP-IKEv2 in scenarios where the same shared secret is used for both. Note that using a shared secret (e.g., a password) in the context EAP-IKEv2 that is identical or similar to a shared secret that is used in another context (including IKEv2) is nevertheless NOT RECOMMENDED.

8.11. Notify Payload

The Notify payload, denoted N(...), is constructed and formatted according to the rules specified in Section 3.10 of [1]. The Protocol ID field of this payload MUST be set to 1 (IKE_SA).

8.12. Next Fast-ID Payload

The Next Fast-ID Payload is defined as follows:

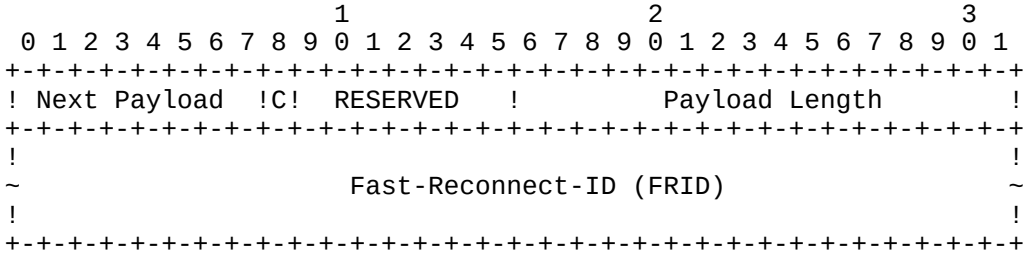

Figure 6: NFID Payload Format

The Next Fast-ID payload, denoted NFID, does not have an equivalent in IKEv2. Nevertheless, the Next Payload, C, RESERVED, and Payload Length fields of this payload are constructed according to Section 3.2 of [1]. The payload ID is registered in Section 11. The Fast-Reconnect-ID field contains a fast reconnect identifier that the peer

can use in the next fast reconnect protocol run, as described in Section 4. In environments where a realm portion is required, Fast-Reconnect-ID includes both a username portion and a realm name portion. The Fast-Reconnect-ID MUST NOT include any terminating null characters. The encoding of the Fast-Reconnect-ID field MUST follow the NAI format [4].

9. Payload Types and Extensibility

In EAP-IKEv2, each payload is identified by means of a type field, which, as specified in [1], is indicated in the "Next Payload" field of the preceding payload. However, the identifier space from which EAP-IKEv2 payload types are drawn is independent from the payload type space of IKEv2. This is because EAP-IKEv2 and IKEv2 may evolve in a different way and, as such, payload types that appear in one protocol do not necessary appear in the other. An example of this is the "Next Fast-ID" (NFID) payload, which does not exist in IKEv2.

The values for the payload types defined in this document are listed in Section 11. Payload type values 13-127 are reserved to IANA for future assignment in EAP-IKEv2. Payload type values 128-255 are for private use among mutually consenting parties.

10. Security Considerations

As mentioned in Section 3, in EAP-IKEv2, the EAP server always assumes the role of the initiator (I), and the EAP peer takes on the role of the responder (R) of an exchange. This is in order to ensure that, in scenarios where the peer authenticates itself based on a password (i.e., in use case 3), operations that involve this password only take place after the server has been successfully authenticated. In other words, this assignment of initiator and responder roles results in protection against offline dictionary attacks on the password that is used by the peer to authenticate itself (see Section 10.7).

In order for two EAP-IKEv2 implementations to be interoperable, they must support at least one common set of cryptographic algorithms. In order to promote interoperability, EAP-IKEv2 implementations MUST support the following algorithms based on the "MUST/MUST-" recommendations given in [5]:

```
Diffie-Hellman Groups: 1024 MODP Group
IKEv2 Transform Type 1 Algorithms: ENCR_3DES
IKEv2 Transform Type 2 Algorithms: PRF_HMAC_SHA1
IKEv2 Transform Type 3 Algorithms: AUTH_HMAC_SHA1_96
```

All other options of [5] MAY be implemented.

The remainder of this section describes EAP-IKEv2 in terms of specific security terminology as required by [2]. The discussion makes reference to the use cases defined in Section 1.

10.1. Protected Ciphersuite Negotiation

In message 3, the EAP server provides the set of ciphersuites it is willing to accept in an EAP-IKEv2 protocol run. Hence, the server is in control of the ciphersuite. An EAP peer that does not support any of the indicated ciphersuites is not able to authenticate. The local security policy of the peer MUST specify the set of ciphersuites that the peer accepts. The server MUST verify that the ciphersuite that is indicated as being chosen by the peer in message 4, belongs to the set of ciphersuites that were offered in message 3. If this verification fails, the server MUST silently discard the packet.

10.2. Mutual Authentication

EAP-IKEv2 supports mutual authentication.

10.3. Integrity Protection

EAP-IKEv2 provides integrity protection of EAP-IKEv2 traffic. This protection is offered after authentication is completed and it is facilitated by inclusion of two Integrity Checksum Data fields: one at the end of the EAP packet (see Figure 4), and one as part of an Encrypted payload (see Section 8.9).

10.4. Replay Protection

EAP-IKEv2 provides protection against replay attacks by a variety of means. This includes the requirement that the Authentication payload is computed as a function of, among other things, a server-provided nonce and a peer-provided nonce. These nonces are required to be practically unpredictable by an adversary. Assuming that the algorithm that is used to compute the Authentication payload does not contain cryptographic weaknesses, the probability that an Authentication payload that is valid in a particular protocol run will also be valid in a subsequent run is therefore negligible.

10.5. Confidentiality

EAP-IKEv2 provides confidentiality of certain EAP-IKEv2 fields, namely those included in Encrypted payloads. With respect to identity confidentiality, the following claims are made. Note that identity confidentiality refers to the EAP-IKEv2 identity of the EAP peer.

Identity confidentiality is provided in the face of a passive adversary, i.e., an adversary that does not modify traffic as it is in transit. Whenever the optional SK{IDr} payload in message 4 of a full EAP-IKEv2 protocol (see Figure 1) is not included, identity confidentiality is also provided in the face of an active adversary. This payload MUST NOT be included in use cases 1, 2, and 3. In use case 4, this payload MUST be included. Therefore, in use case 4, EAP- IKEv2 does not provide identity confidentiality in the face of an active adversary.

Note, however, that the EAP peer provides its identity in message 2 in Figure 1 in cleartext. In order to provide identity confidentiality as discussed in the previous paragraphs, it is necessary to obfuscate the username part of the identity (the realm part must stay intact to allow correct message routing by the Authentication, Authorization, and Accounting (AAA) infrastructure). The EAP server then uses the identity information in message 4. The same mechanism is also used by other EAP methods to provide identity confidentiality, for example, EAP-TTLS [8].

10.6. Key Strength

EAP-IKEv2 supports the establishment of session keys (MSK and EMSK) of a variety of key strengths, with the theoretical maximum at 512 bits per key (since this is the size of the MSK and the EMSK). However, in practice, the effective key strength is likely to be significantly lower, and depends on the authentication credentials used, the negotiated ciphersuite (including the output size of the pseudorandom function), the Diffie-Hellman group used, and on the extent to which the assumptions on which the underlying cryptographic algorithms depend really hold. Of the above mechanisms, the one that offers the lowest key strength can be regarded as a measure of the effective key strength of the resulting session keys. Note that this holds for other EAP methods, too.

Due to the large variety of possible combinations, no indication of a practical effective key strength for MSK or EMSK is given here. However, those responsible for the deployment of EAP-IKEv2 in a particular environment should consider the threats this environment may be exposed to, and configure the EAP-IKEv2 server and peer policies and authentication credentials such that the established session keys are of a sufficiently high effective key strength.

10.7. Dictionary Attack Resistance

EAP-IKEv2 can be used in a variety of use cases, as explained in Section 1. In some of these uses cases, namely use case 1, 2, and 4, dictionary attacks cannot be launched since no passwords are used.

In use case 3, EAP-IKEv2 provides protection against offline dictionary attacks, since operations that involve the password are executed only after the server has authenticated itself (based on a credential other than a password).

In order to reduce exposure against online dictionary attacks, in use case 3, the server SHOULD provide the capability to log failed peer authentication events, and SHOULD implement a suitable policy in case of consecutive failed peer authentication attempts within a short period of time (such as responding with an EAP-Failure instead of message 5 for a predetermined amount of time).

When passwords are used with method 4 (instead of using a key with high entropy), dictionary attacks are possible, as described in Section 8 of [1]:

> "When using pre-shared keys, a critical consideration is how to assure the randomness of these secrets. The strongest practice is to ensure that any pre-shared key contain as much randomness as the strongest key being negotiated. Deriving a shared secret from a password, name, or other low-entropy source is not secure. These sources are subject to dictionary and social engineering attacks, among others."

Hence, the usage of passwords with mode 4 where the EAP peer and the EAP server rely on a shared secret that was derived from a password is insecure. It is strongly recommended to use mode 3 when passwords are used by the EAP peer.

10.8. Fast Reconnect

EAP-IKEv2 supports a "fast reconnect" mode of operation, as described in Section 4.

10.9. Cryptographic Binding

EAP-IKEv2 is not a tunnel EAP method. Thus, cryptographic binding does not apply to EAP-IKEv2.

10.10. Session Independence

EAP-IKEv2 provides session independence in a number of ways, as follows:

Firstly, knowledge of captured EAP-IKEv2 conversations (i.e., the information that a passive adversary may obtain) does not enable the adversary to compute the Master Session Key (MSK) and Extended Master Session Key (EMSK) that resulted from these conversations. This

holds even in the case where the adversary later obtains access to the server and/or the peer's long-term authentication credentials that were used in these conversations. That is, EAP-IKEv2 provides support for "perfect forward secrecy". However, whether or not this support is made use of in a particular EAP-IKEv2 protocol run, depends on when the peer and the server delete the Diffie-Hellman values that they used in that run, and on whether or not they use fresh Diffie-Hellman values in each protocol run. The discussion in Section 2.12 of [1] applies.

Secondly, an active adversary that does not know the peer's and server's long-term authentication credentials cannot learn the MSK and EMSK that were established in a particular protocol run of EAP-IKEv2, even if it obtains access to the MSK and EMSK that were established in other protocol runs of EAP-IKEv2. This is because the MSK and the EMSK are a function of, among other things, data items that are assumed to be generated independently at random in each protocol run.

10.11. Fragmentation

EAP-IKEv2 provides support for fragmentation, as described in Section 8.1.

10.12. Channel Binding

Channel binding is not supported in EAP-IKEv2.

10.13. Summary

EAP security claims are defined in Section 7.2.1 of [2]. The security claims for EAP-IKEv2 are as follows:

Ciphersuite negotiation:	Yes
Mutual authentication:	Yes
Integrity protection:	Yes
Replay protection:	Yes
Confidentiality:	Yes
Key derivation:	Yes; see Section 5
Key strength:	Variable
Dictionary attack prot.:	Yes; see Section 10.7
Fast reconnect:	Yes; see Section 4
Crypt. binding:	N/A
Session independence:	Yes; see Section 10.10
Fragmentation:	Yes; see Section 10.11
Channel binding:	No

11. IANA Considerations

IANA has allocated value 49 for the EAP method type indicating EAP-IKEv2. EAP-IKEv2 has already earlier successfully passed Designated Expert Review as mandated by RFC 3748 for IANA allocations.

In addition, IANA has created a new registry for "EAP-IKEv2 Payloads", and populated it with the following initial entries listed below.

The following payload type values are used by this document.

Next Payload Type	Value
No Next payload	0
Security Association payload	33
Key Exchange payload	34
Identification payload (when sent by initiator, IDi)	35
Identification payload (when sent by responder, IDr)	36
Certificate payload	37
Certificate Request payload	38
Authentication payload	39
Nonce payload	40
Notification payload	41
Vendor ID payload	43
Encrypted payload	46
Next Fast-ID payload	121
RESERVED TO IANA	1-32, 42, 44-45, 47-120, 122-127
PRIVATE USE	128-255

Payload type values 1-120 match the corresponding payloads in the IKEv2 IANA registry. That is, the EAP-IKEv2 payloads that have been assigned a type value in the range 1-120 have a semantically equivalent payload type in IKEv2, with an identical payload type value. However, there exist payloads types in IKEv2 that do not have a semantically equivalent payload in EAP-IKEv2; this explains the fact that the payload type values 42, 44, and 45 have not been assigned in EAP-IKEv2; these values remain RESERVED TO IANA for this version of EAP-IKEv2.

Payload type values 121-127 are used for EAP-IKEv2 specific payloads, i.e., for payloads that do not have a semantically equivalent payload in IKEv2. Note that this range has been reserved for this purpose in the IKEv2 IANA registry too. This means that the same payload type values will not be used for different things in IKEv2 and EAP-IKEv2 protocols.

Payload type values 122-127 are reserved to IANA for future assignment to EAP-IKEv2-specific payloads. Payload type values 128-255 are for private use among mutually consenting parties.

The semantics of the above-listed payloads is provided in this document (0-127) and refer to IKEv2 when necessary (1-120).

New payload type values with a description of their semantic will be assigned after Expert Review. The expert is chosen by the IESG in consultation with the Security Area Directors and the EMU working group chairs (or the working group chairs of a designated successor working group). Updates can be provided based on expert approval only. A designated expert will be appointed by the Security Area Directors. Based on expert approval it is possible to delete entries from the registry or to mark entries as "deprecated".

Each registration must include the payload type value and the semantic of the payload.

12. Contributors

The authors are grateful to Krzysztof Rzecki, Rafal Mijal, Piotr Marnik, and Pawel Matejski, who, during their implementation of EAP-IKEv2 (see http://eap-ikev2.sourceforge.net/), provided invaluable feedback and identified a number of errors in previous versions of this document.

13. Acknowledgements

The authors also thank Pasi Eronen for his invaluable comments as expert reviewer assigned by the EAP working group chairs Jari Arkko and Bernard Aboba. The authors would also like to thank Guenther Horn, Thomas Otto, Paulo Pagliusi, and John Vollbrecht for their insightful comments and suggestions. The members of the PANA design team; in particular, D. Forsberg and A. Yegin, also provided comments on the initial version of this document. We would like to thank Hugo Krawczyk for his feedback regarding the usage of the password-based authentication.

The authors are grateful to the members of the EAP keying design team for their discussion in the area of the EAP Key Management Framework.

We would like to thank Jari Arkko for his support and for his comments. Finally, we would like to thank Charlie Kaufman, Bernard Aboba, and Paul Hoffman for their comments during IETF Last Call.

14. References

14.1. Normative References

[1] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

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

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

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

[5] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", RFC 4307, December 2005.

14.2. Informative References

[6] Aboba, B. and D. Simon, "PPP EAP TLS Authentication Protocol", RFC 2716, October 1999.

[7] Aboba, B., "Extensible Authentication Protocol (EAP) Key Management Framework", Work in Progress, February 2007.

[8] Funk, P. and S. Blake-Wilson, "EAP Tunneled TLS Authentication Protocol (EAP-TTLS)", Work in Progress, July 2004.

Appendix A. EAP-IKEv2 Protocol Runs with Failed Authentication

This appendix illustrates how authentication failures are handled within EAP-IKEv2. Note that authentication failures only occur in full EAP-IKEv2 protocol runs.

Figure 10 shows the message flow in case the EAP peer fails to authenticate the EAP server.

1. R<-I: EAP-Request/Identity

2. R->I: EAP-Response/Identity(Id)

3. R<-I: EAP-Req (HDR, SAi1, KEi, Ni)

4. R->I: EAP-Res (HDR, SAr1, KEr, Nr, [CERTREQ], [SK{IDr}])

5. R<-I: EAP-Req (HDR, SK {IDi, [CERT], [CERTREQ], [IDr], AUTH})

6. R->I: EAP-Res(HDR, SK {N(AUTHENTICATION_FAILED)})

7. R<-I: EAP-Failure

Figure 10: EAP-IKEv2 with Failed Server Authentication

The difference in the full successful exchange described in Section 3 is that, in message 6, the EAP peer MUST answer the EAP server with an Encrypted payload that contains a Notify payload with the Notify Message Type value set to 24 (AUTHENTICATION_FAILED). In that message, the Message ID field in the EAP-IKEv2 header (HDR) MUST carry Message ID value 2. In message 7, an EAP-Failure message MUST be returned by the EAP server.

Figure 11 shows the message flow in case the EAP server fails to authenticate the EAP peer.

1. R<-I: EAP-Request/Identity

2. R->I: EAP-Response/Identity(Id)

3. R<-I: EAP-Req (HDR, SAi1, KEi, Ni)

4. R->I: EAP-Res (HDR, SAr1, KEr, Nr, [CERTREQ], [SK{IDr}])

5. R<-I: EAP-Req (HDR, SK {IDi, [CERT], [CERTREQ], AUTH})

6. R->I: EAP-Res (HDR, SK {IDr, [CERT], AUTH})

7. R<-I: EAP-Req (HDR, SK {N(AUTHENTICATION_FAILED)})

8. R->I: EAP-Res (HDR, SK {})

9. R<-I: EAP-Failure

Figure 11: EAP-IKEv2 with Failed Peer Authentication

Compared to the full successful exchange, one additional roundtrip is required. In message 7, the EAP server MUST send an EAP request with Encrypted payload that contains a Notify payload with the Notify Message Type value set to 24 (AUTHENTICATION_FAILED), instead of sending an EAP-Success message. The EAP peer, upon receiving message 7, MUST send an empty EAP-IKEv2 (informational) message in reply to the EAP server's error indication, as shown in message 8. In messages 7 and 8, the Message ID field in the EAP-IKEv2 header (HDR) MUST carry Message ID value 2. Finally, by means of message 9, the EAP server answers with an EAP-Failure.

Authors' Addresses

Hannes Tschofenig
Nokia Siemens Networks
Otto-Hahn-Ring 6
Munich, Bavaria 81739
Germany

EMail: Hannes.Tschofenig@nsn.com
URI: http://www.tschofenig.com

Dirk Kroeselberg
Nokia Siemens Networks
Otto-Hahn-Ring 6
Munich, Bavaria 81739
Germany

EMail: Dirk.Kroeselberg@nsn.com

Andreas Pashalidis
NEC
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

EMail: pashalidis@nw.neclab.eu

Yoshihiro Ohba
Toshiba America Research, Inc.
1 Telcordia Drive
Piscataway, NJ 08854
USA

EMail: yohba@tari.toshiba.com

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

EMail: florent.ftrd@gmail.com

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

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, THE IETF TRUST 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.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.