Network Working Group
Request for Comments: 4763
Category: Informational

M. Vanderveen
H. Soliman
Qualcomm Flarion Technologies

November 2006

Extensible Authentication Protocol Method for Shared-secret Authentication and Key Establishment (EAP-SAKE)

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

IESG Note

This RFC is not a candidate for any level of Internet Standard. The IETF disclaims any knowledge of the fitness of this RFC for any purpose and in particular notes that the decision to publish is not based on IETF review for such things as security, congestion control, or inappropriate interaction with deployed protocols. The RFC Editor has chosen to publish this document at its discretion. Readers of this document should exercise caution in evaluating its value for implementation and deployment. See RFC 3932 for more information.

Abstract

This document specifies an Extensible Authentication Protocol (EAP) mechanism for Shared-secret Authentication and Key Establishment (SAKE). This RFC is published as documentation for the IANA assignment of an EAP Type for a vendor's EAP method per RFC 3748. The specification has passed Designated Expert review for this IANA assignment.

Table of Contents

1. Introduction

The Extensible Authentication Protocol (EAP), described in [EAP], provides a standard mechanism for support of multiple authentication methods. EAP is also used within IEEE 802 networks through the IEEE 802.11i [IEEE802.11i] framework.

EAP supports several authentication schemes, including smart cards, Kerberos, Public Key, One Time Passwords, TLS, and others. This document defines an authentication scheme, called the EAP-SAKE. EAP-SAKE supports mutual authentication and session key derivation, based on a static pre-shared secret data. This RFC is published as documentation for the IANA assignment of an EAP Type for a vendor's EAP method per RFC 3748. The specification has passed Designated Expert review for this IANA assignment.

2. Terminology

In this document, several words are used to signify the requirements of the specification. These words are often capitalized. The key words "MUST", "MUST NOT", "REQUIRED", "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [KEYWORDS].

In addition to the terms used in [EAP], this document frequently uses the following terms and abbreviations:

MIC Message Integrity Check

SMS SAKE Master Secret

NAI Network Access Identifier

3. Protocol Description

3.1. Overview and Motivation of EAP-SAKE

The EAP-SAKE authentication protocol is a native EAP authentication method. That is, a stand-alone version of EAP-SAKE outside of EAP is not defined. EAP-SAKE is designed to enable mutual authentication, based on pre-shared keys and well-known public cryptographic algorithms. This method is ideal for subscription-based public access networks, e.g., cellular networks.

EAP-SAKE assumes a long-term or pre-shared secret known only to the Peer and the Server. This pre-shared secret is called the Root Secret. The Root Secret consists of a 16-byte part Root-Secret-A, and 16-byte part Root-Secret-B. The Root-Secret-A secret is reserved for use local to the EAP-SAKE method, i.e., to mutually authenticate the EAP Peer and EAP Server. The Root-Secret-B secret is used to derive other quantities such as the Master Session Key (MSK) and Extended Master Session Key (EMSK). Root-Secret-B and Root-Secret-A MUST be cryptographically separate.

When a Backend Authentication Server is used, the Server typically communicates with the Authenticator using an AAA protocol. The AAA communications are outside the scope of this document.

Some of the advantages of EAP-SAKE are as follows:

- It is based on well-established Bellare-Rogaway mutual authentication protocol.
- It supports key derivation for local EAP method use and for export to other third parties to use them independently of EAP.
- It employs only two request/response exchanges.
- It relies on the (corrected) IEEE 802.11i function for key derivation and message integrity. This way a device implementing a lower-layer secure association protocol compliant with IEEE 802.11i standard will not need additional cryptographic code.
- Its encryption algorithm is securely negotiated (with no extra messages), yet encryption use is optional.
- Keys used for authentication and those used for encryption are cryptographically separate.
- User identity anonymity can be optionally supported.

- o No synchronization (e.g., of counters) needed between server and peer.
- o There is no one-time key pre-processing step.

3.2. Protocol Operation

EAP-SAKE uses four messages consisting of two EAP request/response exchanges. The EAP.Success and EAP.Failure messages shown in the figures are not part of the EAP-SAKE method. As a convention, method attributes are named AT_XX, where XX is the name of the parameter the attribute value is set to.

3.2.1. Successful Exchange

A successful EAP-SAKE authentication exchange is depicted in Figure 1. The following steps take place:

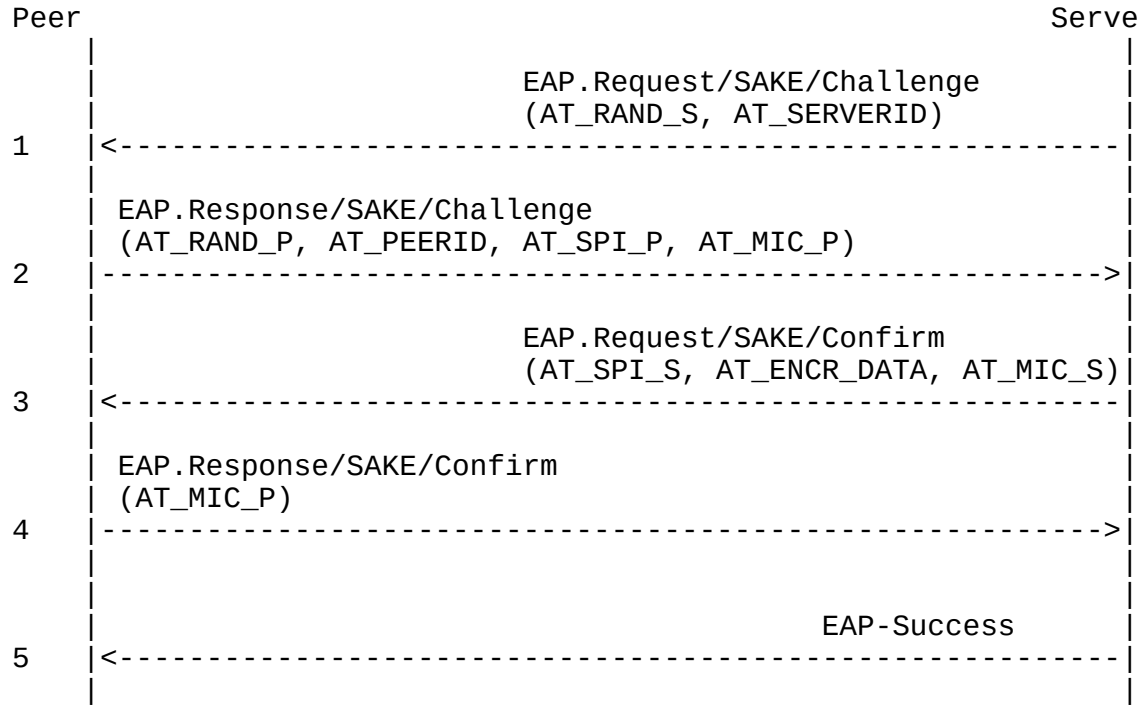


Figure 1. EAP-SAKE Authentication Procedure (with ciphersuite negotiation)

1. The EAP server sends the first message of the EAP-SAKE exchange. This message is an EAP.Request message of type SAKE and subtype Challenge. The EAP.Request/SAKE/Challenge message contains the attributes: AT_RAND_S, whose value is a nonce freshly generated

by the Server; and AT_SERVERID, which carries an identifier of the Server (a fully qualified domain name, such as the realm of the user NAI [NAI]). The AT_SERVERID attribute is OPTIONAL but SHOULD be included in this message. The purpose of the AT_SERVERID is to aid the Peer in selecting the correct security association (i.e., Root-Secret, PEERID, and SERVERID) to use during this EAP phase.

2. The Peer responds by sending an EAP.Response message of type SAKE and subtype Challenge. The EAP.Response/SAKE/Challenge message contains the attributes: AT_RAND_P, which carries a nonce freshly generated by the Peer; AT_PEERID, which carries a Peer identifier; AT_SPI_P, which carries a list of one or more ciphersuites supported by the Peer; and AT_MIC_P, containing a message integrity check. The AT_SPI_P and AT_PEERID attributes are OPTIONAL. The AT_PEERID attribute SHOULD be included, in order to cover the case of multi-homed hosts. A supported ciphersuite is represented by a value local to the EAP-SAKE method, or "Security Parameter Index", see section 3.2.8.3. The Peer uses both nonces, along with the Root-Secret-A key, to derive a SAKE Master Secret (SMS) and Temporary EAP Keys (TEKs), which also include the TEK-Auth and TEK-Cipher keys. The MIC_P value is computed based on both nonces RAND_S and RAND_P, and the entire EAP packet, using the key TEK-Auth, see Section 3.2.6.

3. Upon receipt of the EAP.Response/SAKE/Challenge message, the Server uses both nonces RAND_S and RAND_P, along with the Root-Secret-A key, to compute the SMS and TEK in exactly the same way the Peer did. Following that, the Server computes the Peer’s MIC_P in exactly the same way the Peer did. The Server then compares the computed MIC_P with the MIC_P it received from the Peer. If they match, the Server considers the Peer authenticated. If encryption is needed, the Server selects the strongest ciphersuite from the Peer’s ciphersuite list SPI_P, or a suitable ciphersuite if the Peer did not include the AT_SPI_P attribute. The Server sends an EAP.Request message of type SAKE and subtype Confirm, containing the attributes: AT_SPI_S, carrying the ciphersuite chosen by the Server; AT_ENCR_DATA, containing encrypted data; and AT_MIC_S, carrying a message integrity check. The AT_SPI_S and AT_ENCR_DATA are OPTIONAL attributes, included if confidentiality and/or user identity anonymity is desired. Other OPTIONAL attributes that MAY be included are AT_NEXT_TMPID, and AT_MSK_LIFE. As before, the MIC_S value is computed using both nonces RAND_S and RAND_P, and the entire EAP packet, using the key TEK-Auth.

4. If the Peer receives the EAP.Request/SAKE/Confirm message indicating successful authentication by the Server, the Peer computes the MIC_S in the same manner as the Server did. The Peer then compares the received MIC_S with the MIC_S it computed. If they match, the Peer considers the Server authenticated, and it sends an EAP.Response message of type SAKE and subtype Confirm, with the attribute AT_MIC_P containing a message integrity check, computed in the same manner as before.

5. After a successful EAP-SAKE exchange, the Server concludes the EAP conversation by sending an EAP.Success message to the Peer.

All EAP-SAKE messages contain a Session ID, which is chosen by the Server, sent in the first message, and replicated in subsequent messages until an EAP.Success or EAP.Failure is sent. Upon receipt of an EAP-SAKE message, both Peer and Server MUST check the format of the message to ensure that it is well-formed and contains a valid Session ID.

Note that the Session ID is introduced mainly for replay protection purposes, as it helps uniquely identify a session between a Peer and a Server. In most cases, there is a one-to-one relationship between the Session ID and the Peer/Server pair.

The parameters used by the EAP-SAKE method are summarized in the table below:

Name	Length (bytes)	Description
RAND_P	16	Peer nonce
RAND_S	16	Server nonce
MIC_P	16	Peer MIC
MIC_S	16	Server MIC
SPI_P	variable	Peer ciphersuite preference(s)
SPI_S	variable	Server chosen ciphersuite
PEERID	variable	Peer identifier
SERVERID	variable	Server identifier (FQDN)

3.2.2. Authentication Failure

If the Authenticator receives an EAP.Failure message from the Server, the Authenticator MUST terminate the connection with the Peer immediately.

The Server considers the Peer to have failed authentication if either of the two received MIC_P values does not match the computed MIC_P.

The Server SHOULD deny authorization for a Peer that does not advertise any of the ciphersuites that are considered acceptable (e.g., by local system policy) and send an EAP.Failure message.

In case of authentication failure, the Server MUST send an EAP.Failure message to the Peer as in Figure 2. The following steps take place:

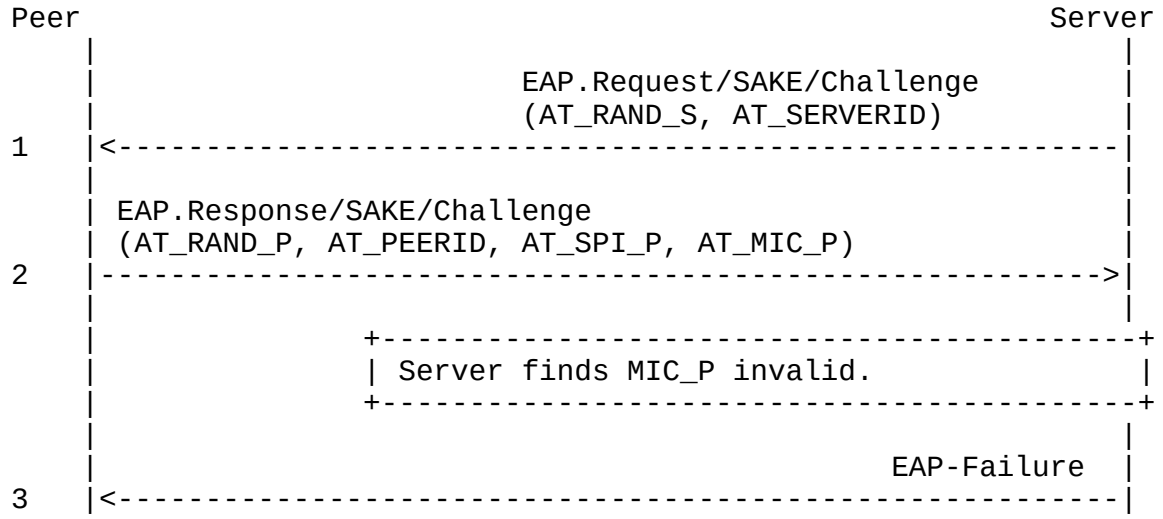


Figure 2. EAP-SAKE Authentication Procedure, Peer Fails Authentication

1. As in step 1 of Figure 1.

2. As in step 2 of Figure 1.

3. Upon receipt of the EAP.Response/SAKE/Challenge message, the Server uses both nonces RAND_S and RAND_P, along with the Root-Secret-A key, to compute the SMS and TEK in exactly the same way the Peer did. Following that, the Server computes the Peer’s MIC in exactly the same way the Peer did. The Server then compares the computed MIC_P with the MIC_P it received from the Peer. Since they do not match, the Server considers the Peer to have failed authentication and sends an EAP.Failure message back to the Peer.

If the AT_SPI_S attribute does not contain one of the SPI values that the Peer listed in the previous message, or if the Peer did not include an AT_SPI_P attribute yet does not accept the ciphersuite the Server has chosen, then the Peer SHOULD silently discard this message. Alternatively, the Peer MAY send a SAKE/Auth-Reject message back to the Server; in response to this message, the Server MUST send an EAP.Failure message to the Peer.

The AT_SPI_S attribute MUST be included if encryption is to be used. The Server knows whether or not encryption is to be used, as it is usually the Server that needs to protect some data intended for the Peer (e.g., temporary ID, group keys, etc). If the Peer receives an AT_SPI_S attribute yet there is no AT_ENCR_DATA attribute, the Peer SHOULD process the message and skip the AT_SPI_S attribute.

The Peer considers the Server to have failed authentication if the received and the computed MIC_S values do not match. In this case, the Peer MUST send to the Server an EAP.Response message of type SAKE and subtype Auth-Reject, indicating an authentication failure. In this case, the Server MUST send an EAP.Failure message to the Peer as in Figure 3. The following steps take place:

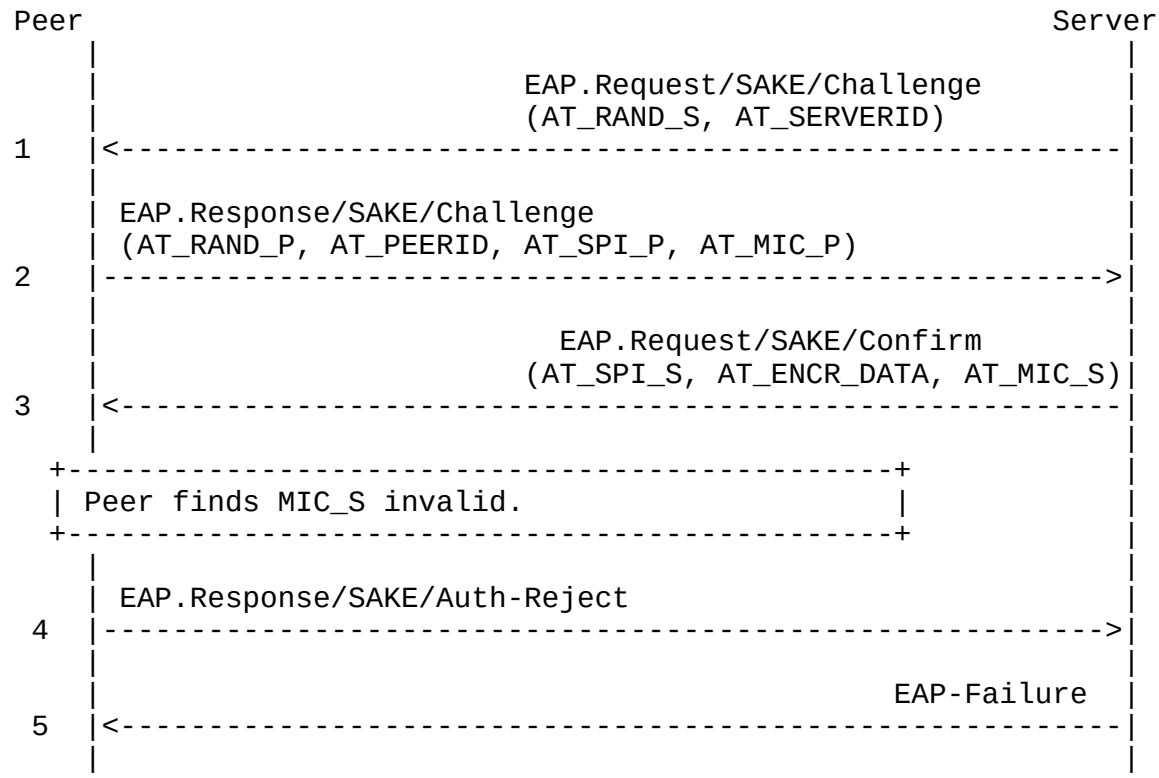


Figure 3. EAP-SAKE Authentication Procedure, Server Fails Authentication

1. As in step 1 of Figure 1.

2. As in step 2 of Figure 1.

3. As in step 3 of Figure 1.

4. The Peer receives a EAP.Request/SAKE/Confirm message indicating successful authentication by the Server. The Peer computes the MIC_S in the same manner as the Server did. The Peer then compares the received MIC_S with the MIC_S it computed. Since they do not match, the Peer considers the Server to have failed authentication. In this case, the Peer responds with an EAP.Response message of type SAKE and subtype Auth-Reject, indicating authentication failure.

5. Upon receipt of a EAP.Response/SAKE/Auth-Reject message, the Server sends an EAP.Failure message back to the Peer.

3.2.3. Identity Management

It may be advisable to assign a temporary identifier (TempID) to a Peer when user anonymity is desired. The TempID is delivered to the Peer in the EAP.Request/SAKE/Confirm message. The TempID follows the format of any NAI [NAI] and is generated by the EAP Server that engages in the EAP-SAKE authentication task with the Peer. EAP servers SHOULD be configurable with TempID spaces that can be distinguished from the permanent identity space. For instance, a specific realm could be assigned for TempIDs (e.g., tmp.example.biz).

A TempID is not assigned an explicit lifetime. The TempID is valid until the Server requests the permanent identifier, or the Peer triggers the start of a new EAP session by sending in its permanent identifier. A Peer MUST be able to trigger an EAP session at any time using its permanent identifier. A new TempID assigned during a successful EAP session MUST replace the existing TempID for future transactions between the Peer and the Server.

Note that the delivery of a TempID does not imply that the Server considers the Peer authenticated; the Server still has to check the MIC in the EAP.Response/SAKE/Confirm message. In case the EAP phase ends with an EAP.Failure message, then the Server and the Peer MUST consider the TempID that was just delivered as invalid. Hence, the Peer MUST NOT use this TempID the next time it tries to authenticate with the Server.

3.2.4. Obtaining Peer Identity

The types of identities that a Peer may possess are permanent and temporary identities, referred to as PermID and TempID, respectively. A PermID can be an NAI associated with the Root Secret, and is long-lived. A TempID is an identifier generated through the EAP method and that the Peer can use to identify itself during subsequent EAP sessions with the Server. The purpose of the TempID is to allow for user anonymity support. The use of TempIDs is optional in the EAP-SAKE method.

The Server MAY request the Peer ID via the EAP.Request/SAKE/Identity message, as shown in Figure 4. This case may happen if, for example, the Server wishes to initiate an EAP-SAKE session for a Peer it does not have a subscriber identifier for. The following steps take place:

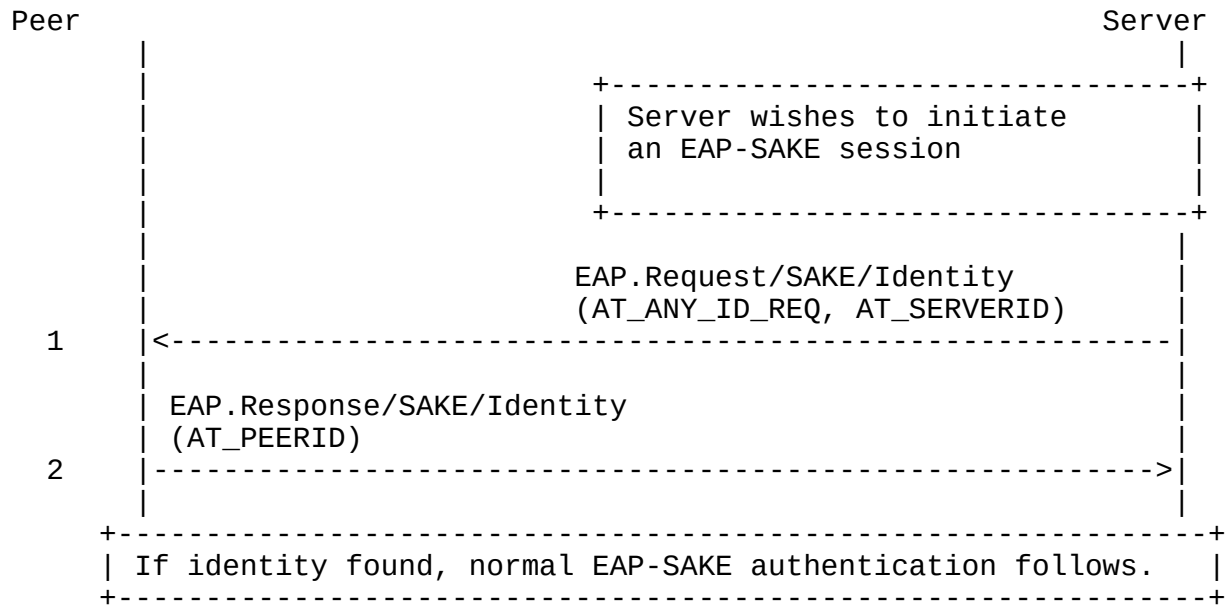


Figure 4. Server Requests Peer Identity

1. The Server sends an EAP.Request message of type SAKE and subtype Identity, with the attribute AT_ANY_ID_REQ, indicating a request for any Peer identifier.

2. The Peer constructs an EAP.Response message of type SAKE and subtype Identity, with the attribute AT_PEER_ID containing any Peer identifier (PermID or TempID).

If the Server cannot find the Peer identity reported in the EAP.Response/SAKE/Identity message, or if it does not recognize the format of the Peer identifier, the Server MAY send an EAP.Failure message to the Peer.

If the Server is unable to look up a Peer by its TempID, or if policy dictates that the Peer should now use its permanent id, then the Server may specifically ask for the permanent identifier, as in Figure 5. The following steps occur:

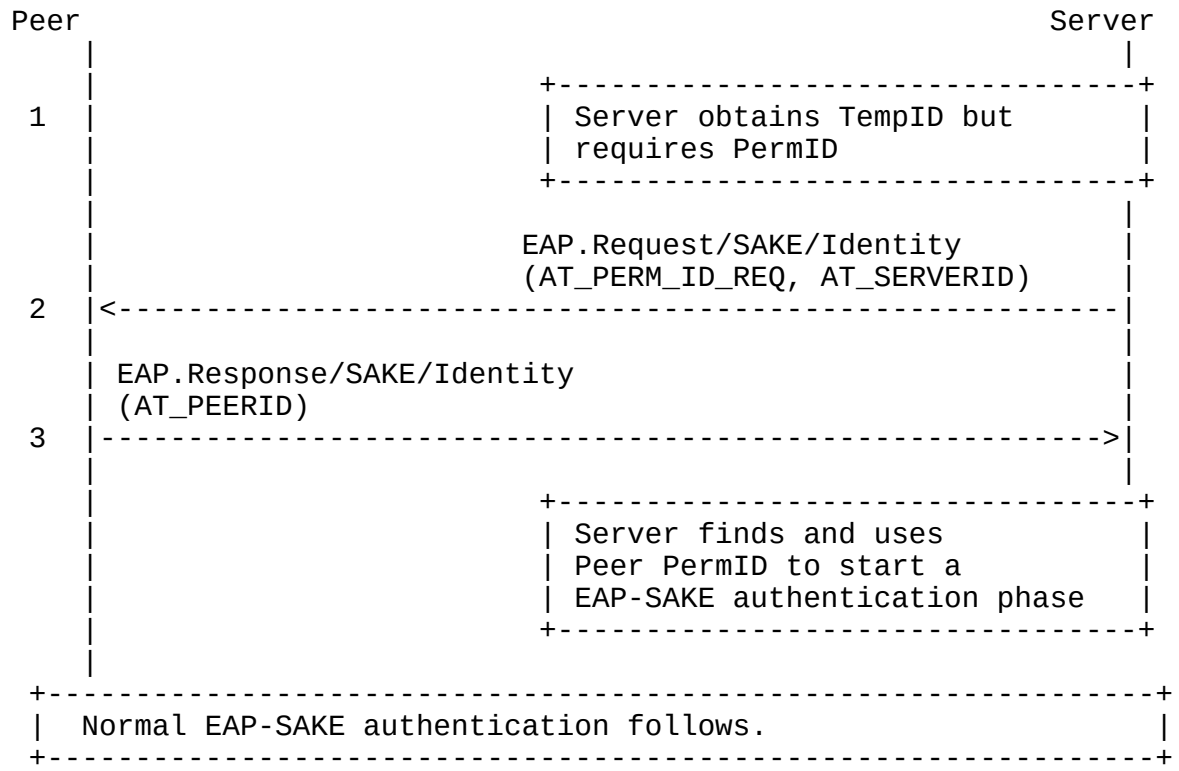


Figure 5. Server Is Given a TempID as Peer Identity; Server Requires a PermID

1. The Peer (or the Authenticator on behalf of the Peer) sends an EAP.Request message of type Identity and includes the TempID.

2. The Server requires a PermID instead, so it sends an EAP.Request message of type SAKE and subtype Identity with attributes AT_PERM_ID_REQ and AT_SERVERID.

3. The Peer sends an EAP.Response message of type SAKE and subtype Identity containing the attribute AT_PEERID carrying the Peer PermID.

3.2.5. Key Hierarchy

EAP-SAKE uses a three-level key hierarchy.

Level 1 contains the pre-shared secret called Root Secret. This is a 32-byte high-entropy string partitioned into the Root-Secret-A part (16 bytes) and the Root-Secret-B part (16 bytes).

Level 2 contains the key derivation key called the SAKE Master Secret (SMS). SMS-A is derived from the Root-Secret-A key and the Peer and Server nonces using the EAP-SAKE Key-Derivation Function (KDF), and similarly for SMS-B. The SMS is known only to the Peer and Server and is not made known to other parties.

Level 3 contains session keys, such as Transient EAP Keys (TEK), Master Session Key (MSK), and Extended MSK (EMSK). TEKs are keys for use local to the EAP method only. They are derived from the SMS-A and the nonces using the EAP-SAKE KDF. They are partitioned into a 16-byte TEK-Auth, used to compute the MICs, and a 16-byte TEK-Cipher, used to encrypt selected attributes. Since the SMS is fresh, so are the derived TEKs.

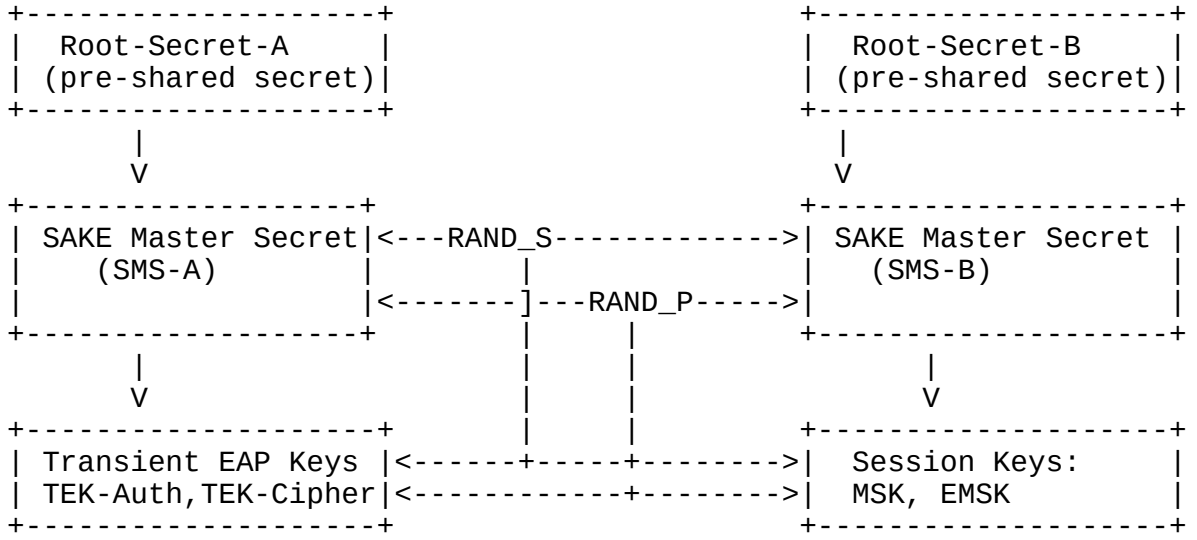


Figure 6. Key Hierarchy for the EAP-SAKE Method

On another branch of level 3 of the key hierarchy are the MSK and EMSK, each mandated to be 64 bytes long. They are derived from the SMS-B and the nonces using the EAP-SAKE KDF. Again, since the SMS is fresh, so are the derived MSK/EMSK. The MSK is meant to be exported and relayed to other parties. The EMSK is reserved for future use, such as derivation of application-specific keys, and is not shared with other parties.

The EAP-SAKE key material is summarized in the table below.

Key	Size (bytes)	Scope	Lifetime	Use
Root-Secret-A	16	Peer, Server	Device	Derive TEK
Root-Secret-B	16	Peer, Server	Device	Derive MSK, EMSK
TEK-Auth	16	Peer, Server	MSK Life	Compute MICs
TEK-Cipher	16	Peer, Server	MSK Life	Encrypt attribute
MSK	64	Peer, Server, Authenticator	MSK Life	Derive keys for lower-layer use
EMSK	64	Peer, Server	MSK Life	Reserved

A key name format is not provided in this version.

Since this version of EAP-SAKE does not support fast re-authentication, the lifetime of the TEKs is to extend only until the next EAP mutual authentication. The MSK lifetime dictates when the next mutual authentication is to take place. The Server MAY convey the MSK lifetime to the Peer in the AT_MSK_LIFE attribute. Note that EAP-SAKE does not support key lifetime negotiation.

The EAP-SAKE Method-Id is the contents of the RAND_S field from the AT_RAND_S attribute, followed by the contents of the RAND_P field in the AT_RAND_P attribute.

3.2.6. Key Derivation

3.2.6.1. Key-Derivation Function

For the rest of this document, KDF-X denotes the EAP-SAKE Key-Derivation Function whose output is X bytes. This function is the pseudo-random function of [IEEE802.11i].

The function takes three strings of bytes of arbitrary lengths: a Key, a Label, and a Msg, and outputs a string Out of length X bytes as follows:

Out = KDF-X (Key, Label, Msg)

The KDF is a keyed hash function with key "Key" and operating on input "Label | Msg". The convention followed herein is that concatenation is denoted by |, FLOOR denotes the floor function, and [x...y] denotes bytes x through y.

The label is an ASCII character string. It is included in the exact form it is given without a length byte or trailing null character.

Below, "Length" denotes a single unsigned octet with values between 0 and 255 (bytes). The following functions are defined (see [HMAC], [SHA1]):

```
H-SHA1(Key, Label, Msg, Length) := HMAC-SHA1(Key, Label|Y|Msg|Length)
where Y := 0x00
KDF-16(Key, Label, Msg) := KDF(Key, Label, Msg, 16)
KDF-32(Key, Label, Msg) := KDF(Key, Label, Msg, 32)
KDF-128(Key, Label, Msg) := KDF(Key, Label, Msg, 128)

KDF(Key, Label, Msg, Length)
R := []  ;; null string
for i := 0 to FLOOR(Length/20)-1 do
R := R | H-SHA1(Key, Label, Msg, i)
return R[0...(Length-1)]
```

3.2.6.2. Transient EAP Keys Derivation

The Peer and Server derive the SMS and then the TEK as follows:

```
SMS-A = KDF-16 (Root-Secret-A, "SAKE Master Secret A", RAND_P|RAND_S)
TEK = KDF-32 (SMS-A, "Transient EAP Key", RAND_S | RAND_P)
TEK-Auth = TEK[0...15]
TEK-Cipher = TEK[16...31]
```

3.2.6.3. Extended/Master Session Key Derivation

The Peer and the Server derive the MSK/EMSK, as follows:

```
SMS-B = KDF-16 (Root-Secret-B, "SAKE Master Secret B", RAND_P|RAND_S)
Session-Key-Block=KDF-128(SMS-B, "Master Session Key", RAND_S|RAND_P)
MSK = Session-Key-Block[0...63]
EMSK = Session-Key-Block[64...127]
```

The derivation above affords the required cryptographic separation between the MSK and EMSK. That is, knowledge of the EMSK does not immediately lead to knowledge of the MSK, nor does knowledge of the MSK immediately lead to knowledge of the EMSK.

3.2.7. Ciphersuite Negotiation

A ciphersuite is identified by a numeric value called the Security Parameter Index (SPI). The SPI is used here in the EAP-SAKE method in order to negotiate a ciphersuite between the Peer and the Server for EAP data protection only. Obviously, this ciphersuite can only be used late in the EAP conversation, after it has been agreed upon by both the Peer and the Server.

The EAP method may or may not need to use this ciphersuite, since attribute encryption is optional. For example, if the temporary identifier needs to be delivered to the Peer and needs to be encrypted, then the negotiated ciphersuite will be used for this task. If there are no attributes that need encryption as they are passed to the Peer, then this ciphersuite is never used.

As with most other methods employing ciphersuite negotiation, the following exchange is employed: the Peer sends an ordered list of one or more supported ciphersuites, starting with the most preferred one, in a field SPI_P. The Server then sends back the one ciphersuite chosen in a field SPI_S. The Server SHOULD choose the strongest ciphersuite supported by both of them.

Ciphersuite negotiation for data protection is achieved via SAKE Signaling as follows. In the EAP.Response/SAKE/Challenge, the Peer sends a list of supported ciphersuites, SPI_P, and protects that with a MIC. In the EAP.Request/SAKE/Confirm, the Server sends one selected ciphersuite, SPI_S, and signs that with a MIC. Finally, the Peer confirms the selected ciphersuite and readiness to use it in a signed EAP.Response/SAKE/Confirm message. The negotiation is secure because of the Message Integrity Checks that cover the entire EAP message.

3.2.8. Message Integrity and Encryption

This section specifies the EAP/SAKE attributes used for message integrity and attribute encryption: AT_MIC_S, AT_MIC_P, AT_IV, AT_ENCR_DATA, and AT_PADDING. Only the AT_MIC_S and AT_MIC_P are mandatory to use during the EAP-SAKE exchange.

Because the TEKs necessary for protection of the attributes and for message authentication are derived using the nonces RAND_S and RAND_P, the AT_MIC_S and AT_MIC_P attributes can only be used in the EAP.Response/SAKE/Challenge message and any messages sent thereafter.

3.2.8.1. The AT_MIC_S and AT_MIC_P Attributes

The AT_MIC_S and AT_MIC_P attributes are used for EAP-SAKE message integrity. The AT_MIC_S attribute MUST be included in all EAP-SAKE packets that the Server sends whenever key material (TEK) has been derived. That is, the AT_MIC_S attribute MUST be included in the EAP.Request/SAKE/Confirm message. The AT_MIC_S MUST NOT be included in EAP.Request/SAKE/Challenge messages, or EAP.Request/SAKE/Identity messages.

The AT_MIC_P attribute MUST be included in all EAP-SAKE packets the Peer sends whenever key material (TEK) has been derived. That is, the AT_MIC_P attribute MUST be included in the EAP.Response/SAKE/Challenge and EAP.Response/SAKE/Confirm messages.

The AT_MIC_P attribute MUST NOT be included in the EAP.Response/SAKE/Auth-Reject message since the Peer has not confirmed that it has the same TEK as the Server.

Messages that do not meet the conditions specified above MUST be silently discarded upon reception.

The value field of the AT_MIC_S and AT_MIC_P attributes contain a message integrity check (MIC). The MIC is calculated over the entire EAP packet, prepended with the Server nonce and identifier and the Peer nonce and identifier. The value field of the MIC attribute is set to zero when calculating the MIC. Including the Server and Peer nonces and identifiers aids in detecting replay and man-in-the-middle attacks.

The Peer computes its MIC as follows:

MIC_P = KDF-16 (TEK-Auth, "Peer MIC", RAND_S | RAND_P | PEERID | 0x00 | SERVERID | 0x00 | <EAP-packet>),

while the Server computes its MIC as

MIC_S = KDF-16 (TEK-Auth, "Server MIC", RAND_P |RAND_S | SERVERID | 0x00 | PEERID | 0x00 | <EAP-packet>).

Here, <EAP-packet> denotes the entire EAP packet, used as a string of bytes, the MIC value field set to zero. 0x00 denotes a single octet value used to delimit SERVERID and PEERID. The PEERID and SERVERID, respectively, are the ones carried in the corresponding attributes in the SAKE/Challenge messages.

In case the SAKE/Challenge exchange was preceded by an EAP.Request/SAKE/Identity message containing the AT_SERVERID Attribute, then the SERVERID value in the MIC_P and MIC_S computation MUST be set to the value of this attribute.

If the AT_SERVERID was not included in either the SAKE/Challenge message or the SAKE/Identity message, then the value of the SERVERID used in the computation of MIC_P and MIC_S MUST be empty. If the AT_PEERID was not included in the SAKE/Challenge message, and there was no EAP.Response/SAKE/Identity message preceding the SAKE/Challenge messages, then the value of the PEERID used in the computation of MIC_P and MIC_S MUST be empty.

The Server and Peer identity are included in the computation of the signed responses so that the Peer and Server can verify each other's identities, and the possession of a common secret, the TEK-Auth. However, since the AT_SERVERID is not explicitly signed with a MIC by the Server, EAP-SAKE does not claim channel binding.

3.2.8.2. The AT_IV, AT_ENCR_DATA, and AT_PADDING Attributes

The AT_IV and AT_ENCR_DATA attributes can be used to transmit encrypted information between the Server and the Peer. The value field of AT_IV contains an initialization vector (IV) if one is required by the encryption algorithm used. It is not mandatory that the AT_IV attribute be included whenever the AT_ENCR_DATA attribute is.

However, the AT_IV attribute MUST NOT be included unless the AT_ENCR_DATA is included. Messages that do not meet this condition MUST be silently discarded.

Attributes can be encrypted only after a ciphersuite has been agreed on, i.e., in any message starting with the Server's EAP.Request/SAKE/Confirm message. The attributes MUST be encrypted using algorithms corresponding to the SPI value specified by the AT_SPI_S attribute. The attributes MUST be encrypted using the TEK-Cipher key, whose derivation is specified in Section 3.2.6.2.

If an IV is required by the encryption algorithm, then the sender of the AT_IV attribute MUST NOT reuse the IV value from previous EAP-SAKE packets. The sender MUST choose a new value for each AT_IV attribute. The sender SHOULD use a good random number generator to generate the initialization vector (see [RFC4086] for guidelines).

The value field of the AT_ENCR_DATA attribute consists of bytes encrypted using the ciphersuite specified in the AT_SPI_S attribute. The encryption key is the TEK-Cipher, and the plaintext consists of one or more concatenated EAP-SAKE attributes.

The default encryption algorithm, as described in Section 3.2.8.3, requires the length of the plaintext to be a multiple of 16 bytes. The sender MAY need to include the AT_PADDING attribute as the last attribute within the value field of the AT_ENCR_DATA attribute. The length of the padding is chosen so that the length of the value field of the AT_ENCR_DATA attribute becomes a multiple of 16 bytes. The AT_PADDING attribute SHOULD NOT be included if the total length of other attributes present within the AT_ENCR_DATA attribute is a multiple of 16 bytes. The length of the AT_PADDING attribute includes the Attribute Type and Attribute Length fields. The actual pad bytes in the value field are set to zero (0x00) on sending. The recipient of the message MUST verify that the pad bytes are zero after decryption and MUST silently discard the message otherwise.

The MIC computed on the entire EAP message can be used to obviate the need for special integrity protection or message authentication of the encrypted attributes.

An example of the AT_ENCR_DATA attribute is shown in Section 3.3.3.

3.2.8.3. Security Parameter Index (SPI) Considerations

An SPI value is a variable-length string identifying at least an encryption algorithm and possibly a "security association". EAP-SAKE does not mandate the format of the SPI; it only mandates that the default encryption algorithm be supported if encryption is supported. That is, if an implementation compliant with this document supports encryption, then it MUST support the AES-CBC cipher.

The default encryption algorithm AES-CBC involves the AES block cipher [AES] in the Cipher-Block-Chaining (CBC) mode of operation [CBC].

The Peer in the EAP-SAKE method can send up to four SPI values in its SPI_P field. Because the length of the AT_SPI_P and AT_SPI_S attributes must each be a multiple of 2 bytes, the sender pads the value field with zero bytes when necessary (the AT_PADDING attribute is not used here). For example, the value field of the AT_SPI_S contains one byte with the chosen SPI, followed by one byte of zeros.

3.2.9. Fragmentation

The EAP-SAKE method does not require fragmentation, as the messages do not get excessively long. That is, EAP packets are well within the limit of the maximum transmission unit of other layers (link, network). The only variable fields are those carrying NAIs, which are limited by their length field to 256 bytes.

3.2.10. Error Cases

Malformed messages: As a general rule, if a Peer or Server receives an EAP-SAKE packet that contains an error, the implementation SHOULD silently discard this packet, not change state, and not send any EAP messages to the other party. Examples of such errors include a missing mandatory attribute, an attribute that is not allowed in this type of message, and unrecognized subtypes or attributes.

Non-matching Session Id: If a Peer or Server receives an EAP-SAKE packet with a Session Id field not matching the Session Id from the previous packet in this session, that entity SHOULD silently discard this packet (not applicable for the first message of an EAP-SAKE session).

Peer Authorization Failure: It may be possible that a Peer is not authorized for services, such as when the terminal device is reported stolen. In that case, the Server SHOULD send an EAP.Failure message.

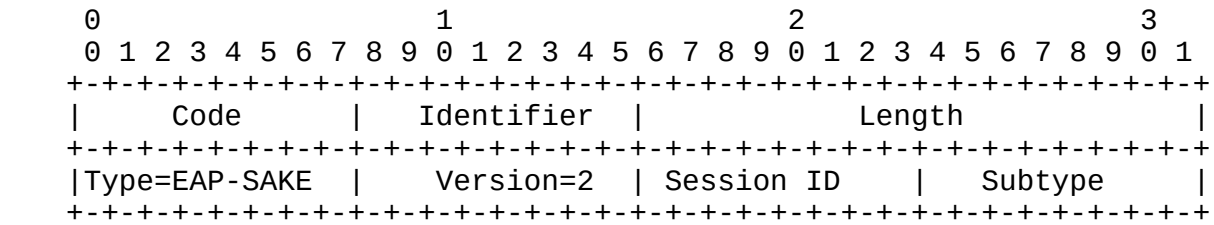
Unexpected EAP.Success: A Server MUST NOT send an EAP-Success message before the SAKE/Challenge and SAKE/Confirm rounds. The Peer MUST silently discard any EAP.Success packets before the Peer has successfully authenticated the Server via the EAP.Response/SAKE/Confirm packet.

The Peer and Server SHOULD log all error cases.

3.3. Message Formats

3.3.1. Message Format Summary

A summary of the EAP packet format [EAP] is shown below for convenience. The fields are transmitted from left to right.



Code

1 - Request

2 - Response

Identifier

The identifier field is one octet and aids in matching responses with requests.

Length

The Length field is two octets and indicates the number of octets in an EAP message including the Code, Identifier, Length, Type, and Data fields.

Type

To be assigned.

Version

The EAP-SAKE method as described herein is version 2.

Session ID

The Session ID is a 1-byte random number that MUST be freshly generated by the Server that must match all EAP messages in a particular EAP conversation.

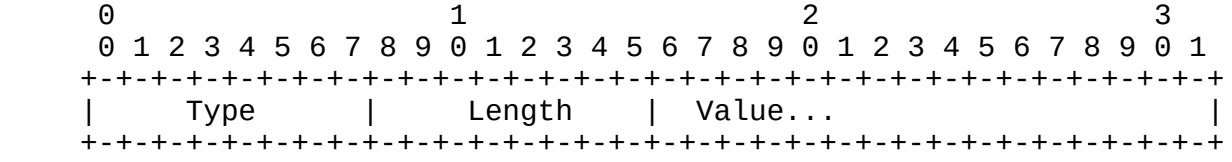
Subtype

EAP-SAKE subtype: SAKE/Challenge, SAKE/Confirm, SAKE/Auth-Reject, and SAKE/Identity. All messages of type "EAP-SAKE" that are not of these subtypes MUST silently discarded.

Message Name	Subtype Value (decimal)
SAKE/Challenge	1
SAKE/Confirm	2
SAKE/Auth-Reject	3
SAKE/Identity	4

3.3.2. Attribute Format

The EAP-SAKE attributes that are part of the EAP-SAKE packet follow the Type-Length-Value format with 1-byte Type, 1-byte Length, and variable-length Value (up to 255 bytes). The Length field is in octets and includes the length of the Type and Length fields. The EAP-SAKE attribute format is as follows:



Type

1-byte unsigned integer; see Table below.

Length

The total number of octets in the attribute, including Type and Length.

Value

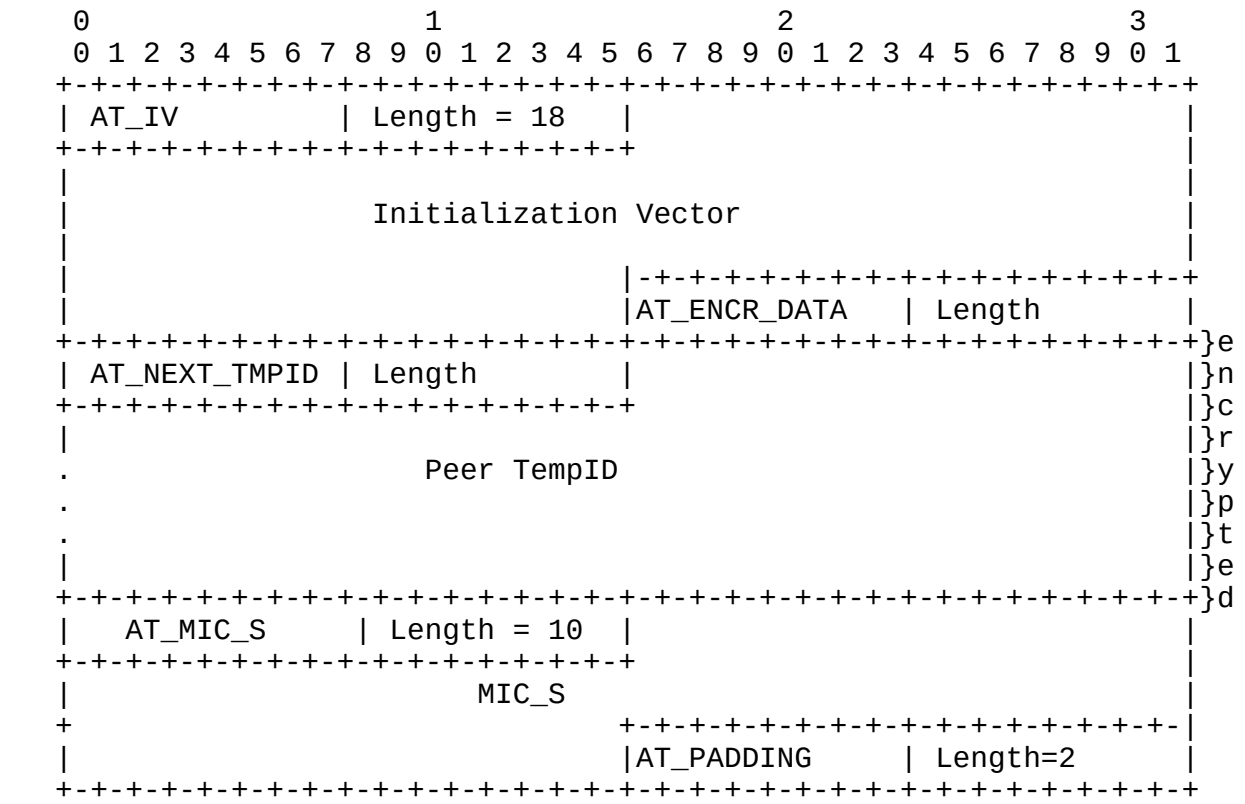
Attribute-specific.

The following attribute types are allocated.

Attr. Name	Length (bytes)	Skippable	Description
AT_RAND_S	18	No	Server Nonce RAND_S
AT_RAND_P	18	No	Peer Nonce RAND_P
AT_MIC_S	10	No	Server MIC
AT_MIC_P	10	No	Peer MIC
AT_SERVERID	variable	No	Server FQDN
AT_PEERID	variable	No	Peer NAI (tmp, perm)
AT_SPI_S	variable	No	Server chosen SPI SPI_S
AT_SPI_P	variable	No	Peer SPI list SPI_P
AT_ANY_ID_REQ	4	No	Requires any Peer Id (tmp, perm)
AT_PERM_ID_REQ	4	No	Requires Peer’s permanent Id/NAI
AT_ENCR_DATA	Variable	Yes	Contains encrypted attributes
AT_IV	Variable	Yes	IV for encrypted attributes
AT_PADDING	2 to 18	Yes	Padding for encrypted attributes
AT_NEXT_TMPID	variable	Yes	TempID for next EAP-SAKE phase
AT_MSK_LIFE	6	Yes	MSK Lifetime in seconds

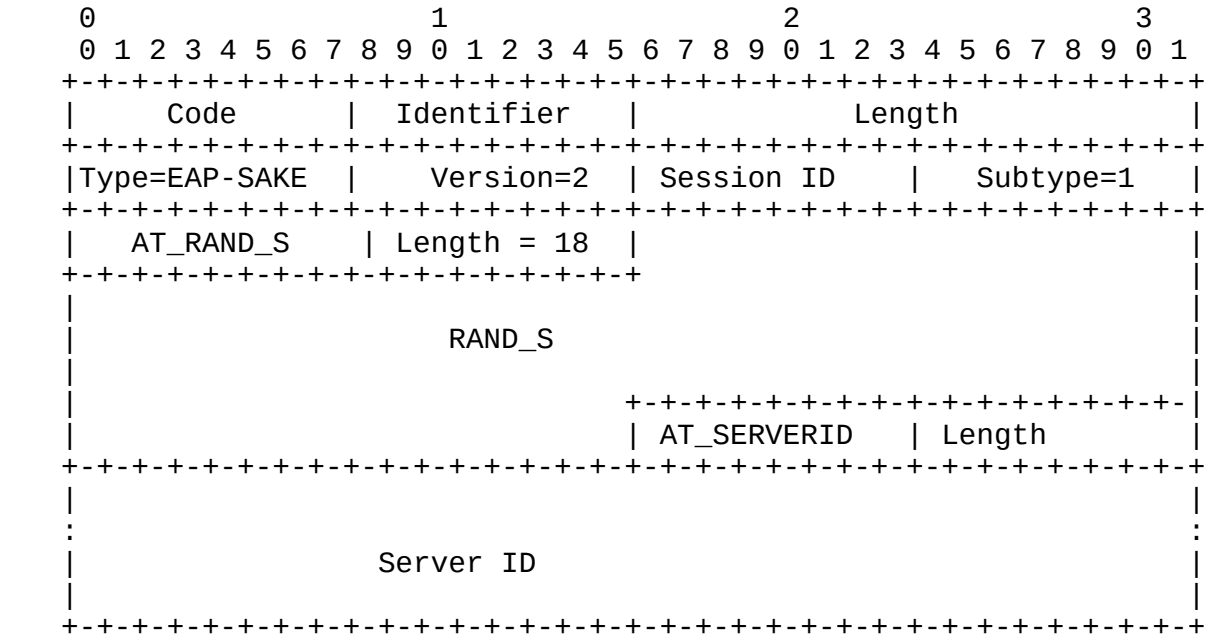
3.3.3. Use of AT_ENCR_DATA Attribute

An example of the AT_ENCR_DATA attribute, as used in the EAP.Request/SAKE/Confirm message, is shown below:



3.3.4. EAP.Request/SAKE/Challenge Format

The format of the EAP.Request/SAKE/Challenge packet is shown below.



The semantics of the fields is described below:

Code

1 for Request

Identifier

A random number. See [EAP].

Length

The length of the entire EAP packet in octets.

Type

EAP-SAKE

Version

2

Session ID

A random number chosen by the server to identify this EAP-Session.

Subtype

1 for SAKE/Challenge

AT_RAND_S

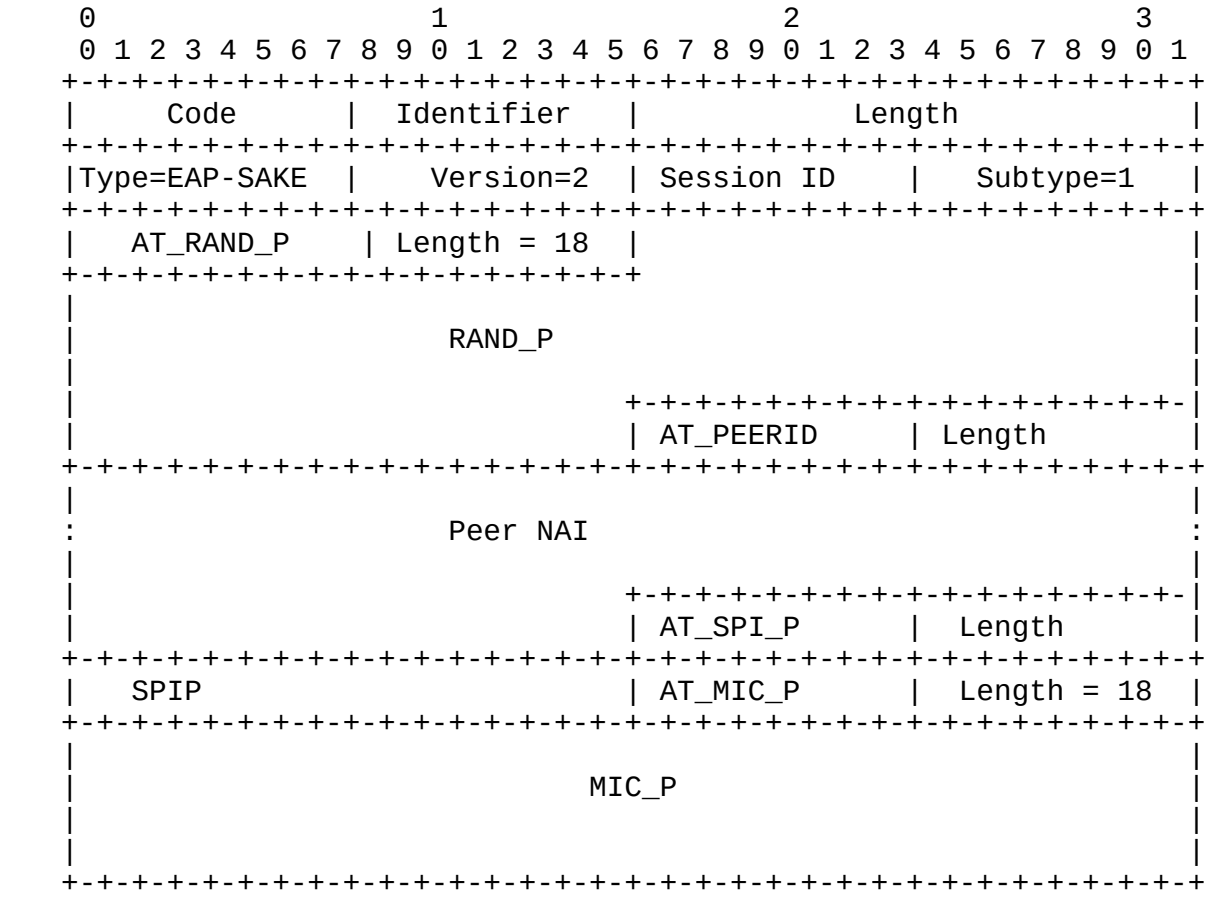
The value field of this attribute contains the Server nonce RAND_S parameter. The RAND_S attribute MUST be present in EAP.Request/SAKE/Challenge.

AT_SERVERID

The value field of this attribute contains the Server identifier (e.g., a non-null terminated string). The AT_SERVERID attribute SHOULD be present in EAP.Request/SAKE Challenge.

3.3.5. EAP.Response/SAKE/Challenge Format

The format of the EAP.Response/SAKE/Challenge packet is shown below.



The semantics of the fields is described below:

Code

2 for Response

Identifier

A number that MUST match the Identifier field from the corresponding Request.

Length

The length of the entire EAP packet in octets.

Type

EAP-SAKE

Version

2

Session ID

A number matching all other EAP messages in this EAP session.

Subtype

1 for SAKE/Challenge

AT_RAND_P

The value field of this attribute contains the Peer nonce RAND_P parameter. The AT_RAND_P attribute MUST be present in the EAP.Response/SAKE/Challenge.

AT_PEERID

The value field of this attribute contains the NAI of the Peer. The Peer identity follows the same Network Access Identifier format that is used in EAP.Response/Identity, i.e., including the NAI realm portion. The identity is the permanent identity, or a temporary identity. The identity does not include any terminating null characters. The AT_PEERID attribute is optional in the EAP.Response/SAKE/Challenge.

AT_SPI_P

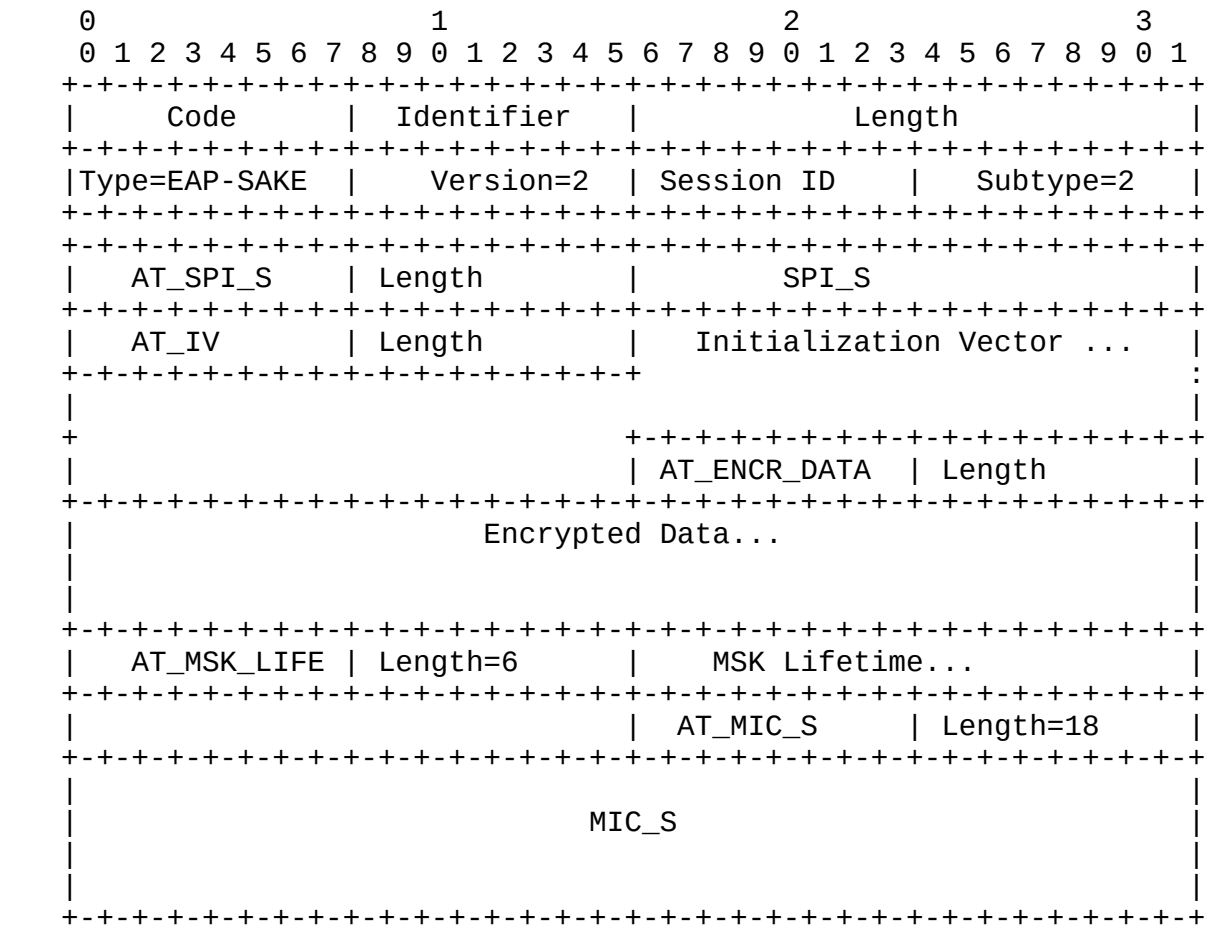
The value field of this attribute contains the Peer's ciphersuite list SPI_P parameter. The AT_SPI_P attribute is optional in the EAP.Response/SAKE/Challenge.

AT_MIC_P

The value field of this attribute contains the Peer MIC_P parameter. The AT_MIC_P attribute MUST be present in the EAP.Response/SAKE/Challenge.

3.3.6. EAP.Request/SAKE/Confirm Format

The format of the EAP.Request/SAKE/Confirm packet is shown below.



The semantics of the fields is described below:

Code

1 for Request

Identifier

A random number. See [EAP].

Length

The length of the entire EAP packet in octets.

Type

EAP-SAKE

Version

2

Session ID

A number matching all other EAP messages in this EAP session.

Subtype

2 for SAKE Confirm

AT_SPI_S

The value field of this attribute contains the Server chosen ciphersuite SPI_S parameter. The AT_SPI_S attribute is optional in the EAP.Request/SAKE/Confirm.

AT_IV

This attribute is optional to use in this message. The value field of this attribute contains the Initialization Vector that is used with the encrypted data following.

AT_ENCR_DATA

This attribute is optional to use in this message. The encrypted data, if present, may contain an attribute AT_NEXT_TMPID, containing the NAI the Peer should use in the next EAP authentication.

AT_MSK_LIFE

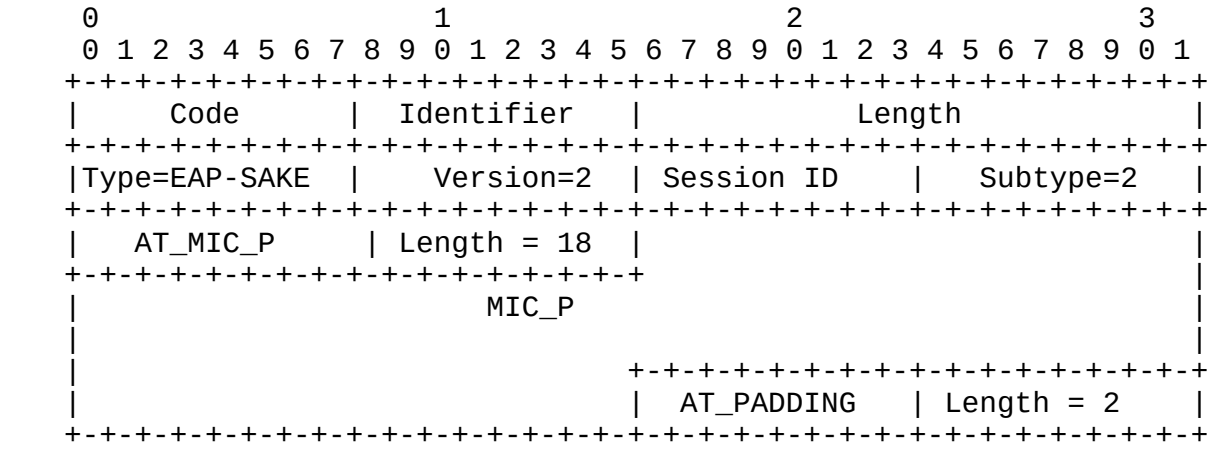
This attribute is optional to use in this message. The value field of this attribute contains the MSK Lifetime in seconds.

AT_MIC_S

The value field of this attribute contains the Server MIC_S parameter. The AT_MIC_S attribute MUST be present in the EAP.Request/SAKE/Confirm.

3.3.7. EAP.Response/SAKE/Confirm Format

The format of the EAP.Response/SAKE/Confirm packet is shown below.



The semantics of the fields is described below:

Code

2 for Response

Identifier

A number that MUST match the Identifier field from the corresponding Request.

Length

The length of the entire EAP packet in octets.

Type

EAP-SAKE

Version

2

Session ID

A number matching all other EAP messages in this EAP session.

Subtype

2 for SAKE Confirm

AT_MIC_P

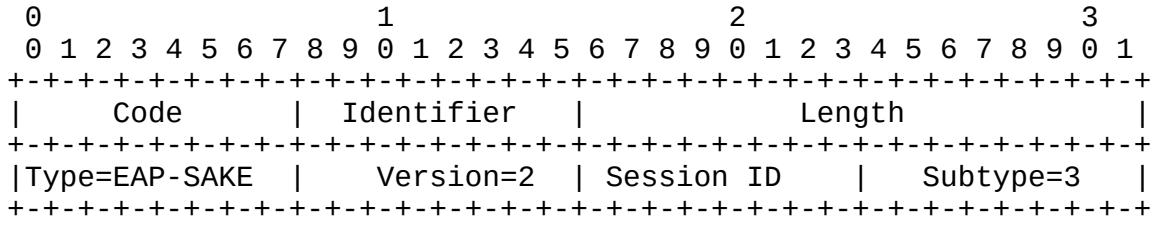
The value field of this attribute contains the Peer's MIC_P parameter. The AT_MIC_P attribute MUST be present in the EAP.Response/SAKE/Confirm.

AT_PADDING

The value field is set to zero. Added to achieve 32-bit alignment of the EAP-SAKE packet.

3.3.8. EAP.Response/SAKE/Auth-Reject Format

The format of the EAP.Response/SAKE/Auth-Reject packet is shown below.



The semantics of the fields is described below:

Code

2 for Response

Identifier

A number that MUST match the Identifier field from the corresponding Request.

Length

The length of the entire EAP packet in octets.

Type

EAP-SAKE

Version

2

Session ID

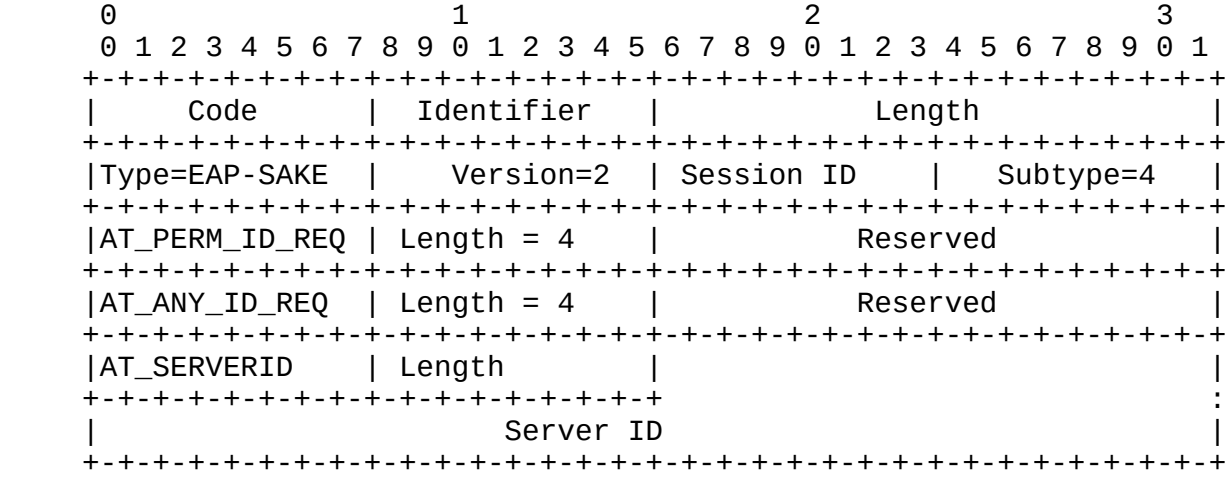
A number matching all other EAP messages in this EAP session.

Subtype

3 for SAKE/Auth-Reject

3.3.9. EAP.Request/SAKE/Identity Format

The format of the EAP.Request/SAKE/Identity is shown below.



The semantics of the fields is described below:

Code

1 for Request

Identifier

A random number. See [EAP].

Length

The length of the entire EAP packet in octets.

Type

EAP-SAKE

Version

2

Session ID

A number matching all other EAP messages in this EAP session.

Subtype

4 for SAKE/Identity

AT_PERM_ID_REQ

The AT_PERM_ID_REQ attribute is optional, to be included in cases where the Server requires the Peer to give its permanent identifier (i.e., PermID). The AT_PERM_ID_REQ MUST NOT be included if the AT_ANY_ID_REQ attribute is included. The value field only contains two reserved bytes, which are set to zero on sending and ignored on reception.

AT_ANY_ID_REQ

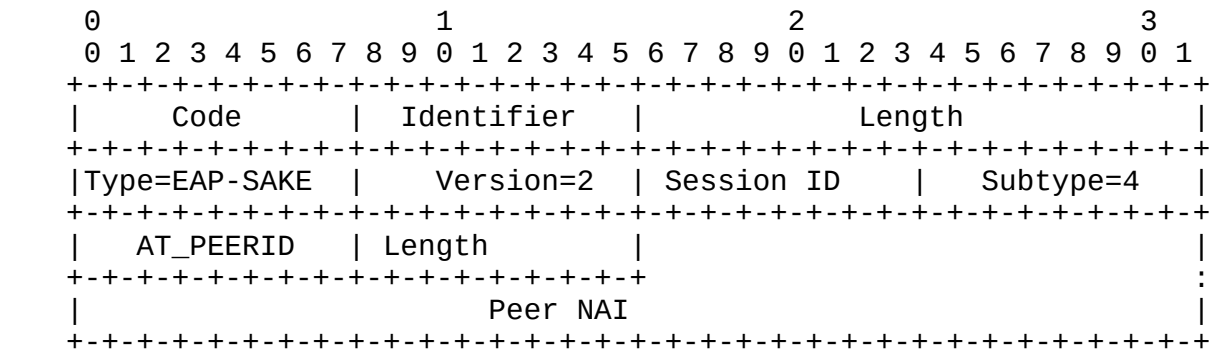
The AT_ANY_ID_REQ attribute is optional, to be included in cases where the Server requires the Peer to send any identifier (e.g., PermID, TempID). The AT_ANY_ID_REQ MUST NOT be included if AT_PERM_ID_REQ is included. The value field only contains two reserved bytes, which are set to zero on sending and ignored on reception. One of the AT_PERM_ID_REQ and AT_ANY_ID_REQ MUST be included.

AT_SERVERID

The value field of this attribute contains the identifier/realm of the Server. The AT_SERVERID attribute is optional but RECOMMENDED to include in the EAP.Request/SAKE/Identity.

3.3.10. EAP.Response/SAKE/Identity Format

The format of the EAP.Response/SAKE/Identity is shown below:



The semantics of the fields is described below:

Code

2 for Response

Identifier

A number that MUST match the Identifier field from the corresponding Request.

Length

The length of the entire EAP packet.

Type

EAP-SAKE

Version

2

Session ID

A number matching all other EAP messages in this EAP session.

Subtype

4 for SAKE/Identity

AT_PEERID

The value field of this attribute contains the NAI of the Peer. The AT_PEERID attribute MUST be present in EAP.Response/SAKE/Identity.

3.3.11. Other EAP Messages Formats

The format of the EAP.Request/Identity and EAP.Response/Identity packets is described in [EAP]. The user ID (e.g., NAI) SHOULD be present in this packet.

The format of the EAP-Success and EAP-Failure packet is also shown in [EAP].

4. IANA Considerations

IANA allocated a new EAP Type for EAP-SAKE.

EAP-SAKE messages include an 8-bit Subtype field. The Subtype is a new numbering space for which IANA administration is required. The following subtypes are specified in this memo:

SAKE/Challenge.....1
SAKE/Confirm.....2
SAKE/Auth-Reject.....3
SAKE/Identity.....4

The Subtype-specific data is composed of attributes, which have an 8-bit type number. Attributes numbered within the range 0 through 127 are called non-skippable attributes, and attributes within the range of 128 through 255 are called skippable attributes. The EAP-SAKE attribute type number is a new numbering space for which IANA administration is required. The following attribute types are specified:

AT_RAND_S.....1
AT_RAND_P.....2
AT_MIC_S.....3
AT_MIC_P.....4
AT_SERVERID.....5
AT_PEERID.....6
AT_SPI_S.....7
AT_SPI_P.....8
AT_ANY_ID_REQ.....9
AT_PERM_ID_REQ.....10

AT_ENCR_DATA.....128
AT_IV.....129
AT_PADDING.....130
AT_NEXT_TMPID.....131
AT_MSK_LIFE.....132

All requests for value assignment from the two number spaces described in this memo require proper documentation, according to the "Specification Required" policy described in [IANA].

All assignments of values from the two number spaces described in this memo require IETF consensus.

5. Security Considerations

The EAP specification [EAP] describes the security vulnerabilities of EAP, which does not include its method-specific security mechanisms. This section discusses the claimed security properties of the EAP-SAKE method, along with vulnerabilities and security recommendations.

5.1. Denial-of-Service Attacks

Since EAP-SAKE is not a tunneling method, the EAP.Response/SAKE/Auth-Reject, EAP.Success, and EAP.Failure packets are not integrity or replay protected. This makes it possible for an attacker to spoof such messages. Note that EAP.Response/SAKE/Auth-Reject cannot be protected with a MIC since an authentication failure indicates that the Server and Peer do not agree on a common key.

Most importantly, an attacker cannot cause a Peer to accept an EAP.Success packet as indication that the Server considers the mutual authentication to have been achieved. This is because a Peer does not accept EAP.Success packets before it has authenticated the Server or after it has considered the Server to have failed authentication.

5.2. Root Secret Considerations

If the Root Secret is known to any party other than the Server and Peer, then the mutual authentication and key establishment using EAP-SAKE is compromised.

EAP-SAKE does not address how the Root Secret is generated or distributed to the Server and Peer. It is RECOMMENDED that the entropy of the Root Secret be maximized. The Root Secret SHOULD be machine-generated.

If the Root Secret is derived from a low-entropy, guessable quantity such as a human-selected password, then the EAP-SAKE key derivation is subject to on-line and off-line dictionary attacks. To help identify whether such a password has been compromised, implementations SHOULD keep a log of the number of EAP-SAKE messages received with invalid MIC fields. In these cases, a procedure for updating the Root Secret securely SHOULD be in place.

5.3. Mutual Authentication

Mutual authentication is accomplished via the SAKE/Challenge and SAKE/Confirm messages. The EAP.Request/SAKE/Challenge contains the Server nonce RAND_S; the EAP.Response/SAKE/Challenge contains the Peer nonce RAND_P, along with the Peer MIC (MIC_P); and the EAP.Request/SAKE/Confirm contains the Server MIC (MIC_S). Both MICs (MIC_S and MIC_P) are computed using both nonces RAND_S and RAND_P and are keyed by the TEK, a shared secret derived from the Root Secret. The Server considers the Peer authenticated if the MIC_P it computes matches the one that the Peer sends. Similarly, the Peer considers the Server authenticated if the MIC_S it computes matches the one that the Server sends. The way the MICs are computed involves a keyed one-way hash function, which makes it computationally hard for an attacker to produce the correct MIC without knowledge of the shared secret.

5.4. Integrity Protection

Integrity protection of EAP-SAKE messages is accomplished through the use of the Message Integrity Checks (MIC), which are present in every message as soon as a common shared secret (TEK) is available, i.e., any message after the EAP.Request/SAKE/Challenge. An adversary cannot modify any of the MIC-protected messages without causing the recipient to encounter a MIC failure. The extent of the integrity protection is commensurate with the security of the KDF used to derive the MIC, the length and entropy of the shared secret used by the KDF, and the length of the MIC.

5.5. Replay Protection

The first message of most session establishment protocols, such as EAP-SAKE, is subject to replay. A replayed EAP.Request/SAKE/Challenge message results in the Peer sending an EAP.Response/SAKE/Challenge message back, which contains a MIC that was computed using the attacker’s chosen nonce. This poses a minimal risk to the compromise of the TEK-Auth key, and this EAP Session cannot proceed successfully as the Peer will find the Server’s MIC invalid.

Replay protection is achieved via the RAND_S and RAND_P values, together with the Session ID field, which are included in the calculation of the MIC present in each packet subsequent to the EAP-SAKE/Challenge request packet. The Session ID MUST be generated anew by the Server for each EAP session. Session Ids also aid in identification of possible multiple EAP sessions between a Peer and a Server. Within the same session, messages can be replayed by an attacker, but the state machine SHOULD be able to handle these cases. Note that a replay within a session is indistinguishable to a recipient from a network malfunction (e.g., message was first lost and then re-transmitted, so the recipient thinks it is a duplicate message).

Replay protection between EAP sessions and within an EAP session is also accomplished via the MIC, which covers not only the entire EAP packet (including the Session ID) but also the nonces RAND_S and RAND_P. Thus, the recipient of an EAP message can be assured that the message it just received is the one just sent by the other Peer and not a replay, since it contains a valid MIC of the recipient's nonce and the other Peer nonce. As before, the extent of replay protection is commensurate with the security of the KDF used to derive the MIC, the length and entropy of the shared secret used by the KDF, and the length of the MIC.

5.6. Confidentiality

Confidentiality of EAP-SAKE attributes is supported through the use of the AT_ENCR_DATA and AT_IV attributes. A ciphersuite is negotiated securely (see Section 3.2.7) and can be used to encrypt any attributes as needed. The default ciphersuite contains a strong cipher based on AES.

5.7. Key Derivation, Strength

EAP-SAKE derives a Master Key (for EAP use) and Master Session Key, as well as other lower-level keys, such as TEKs. Some of the lower-level keys may or may not be used. The strength (entropy) of all these keys is at most the strength of the Root Secret.

The entropy of the MSK and of the EMSK, assuming that the Server and Peer 128-bit nonces are generated using good random number generators, is at most 256-bits.

5.8. Dictionary Attacks

Dictionary attacks are not feasible to mount on the EAP-SAKE method because passwords are not used. Instead, the Root Secret is machine-generated. This does not necessarily pose provisioning problems.

5.9. Man-in-the-Middle Attacks

Resistance to man-in-the-middle attacks is provided through the integrity protection that each EAP message carries (i.e., Message Integrity Check field) as soon as a common key for this EAP session has been derived through mutual authentication. As before, the extent of this resistance is commensurate with the strength of the MIC itself. Man-in-the-middle attacks associated with the use of any EAP method within a tunneling or sequencing protocol are beyond the scope of this document.

5.10. Result Indication Protection

EAP-SAKE provides result indication protection in that it provides result indications, integrity protection, and replay protection. The Server indicates that it has successfully authenticated the Peer by sending the EAP.Request/SAKE/Confirm message, which is integrity and replay protected. The Peer indicates that it has successfully authenticated the Server by sending the EAP.Response/SAKE/Confirm message, which is also integrity and replay protected.

5.11. Cryptographic Separation of Keys

The TEKs used to protect EAP-SAKE packets (TEK-Auth, TEK-Cipher), the Master Session Key, and the Extended Master Session Key are cryptographically separate. Information about any of these keys does not lead to information about any other keys. We also note that it is infeasible to calculate the Root Secret from any or all of the TEKs, the MSK, or the EMSK.

5.12. Session Independence

Within each EAP-SAKE session, fresh keying material is generated. The keying material exported by this method from two independent EAP-SAKE sessions is cryptographically separate, as explained below.

Both the Server and the Peer SHOULD generate fresh random numbers (i.e., nonces) for the EAP-SAKE exchange. If either entity re-uses a random number from a previous session, then the fact that the other does use a freshly generated random number implies that the TEKs, MSK, and EMSK derived within this session are cryptographically

separate from the corresponding keys derived in the previous exchange.

Therefore, compromise of MSK or EMSK on one exchange does not compromise the MSK and EMSK of previous or subsequent exchanges between a Peer and a Server.

5.13. Identity Protection

As seen from Section 3.2.3., the Server may assign a temporary NAI to a Peer in order to achieve user anonymity. This identifier may be used by the Peer the next time it engages in an EAP-SAKE authentication phase with the Server. The TempID is protected by sending it encrypted, within an AT_ENCR_DATA attribute, and signed by the Server with a MIC. Thus, an eavesdropper cannot link the original PermID that the Peer first sends (e.g., on power-up) to any subsequent TempID values sent in the clear to the Server.

The Server and Peer MAY be configured such that only TempID identities are exchanged after one initial EAP-SAKE phase that uses the Peer permanent identity. In this case, in order to achieve maximum identity protection, the TempID SHOULD be stored in non-volatile memory in the Peer and Server. Thus, compliance with this document does not preclude or mandate Peer identity protection across the lifetime of the Peer.

5.14. Channel Binding

The Server identifier and Peer identifier MAY be sent in the SAKE/Challenge messages. However, since there is no established authentication key at the time of the first message, the Server identifier is not integrity-protected here.

All subsequent EAP-SAKE messages exchanged during a successful EAP-SAKE phase are integrity-protected, as they contain a Message Integrity Check (MIC). The MIC is computed over the EAP message and also over the Server and Peer identities. In that, both EAP endpoints can verify the identity of the other party.

5.15. Ciphersuite Negotiation

EAP-SAKE does not support negotiation of the ciphersuite used to integrity-protect the EAP conversation. However, negotiation of a ciphersuite for data protection is supported. This ciphersuite negotiation is protected in order to minimize the risk of down-negotiation or man-in-the-middle attacks.

This negotiation is secure because of the Message Integrity Checks (MICs) that cover the entire EAP messages used for ciphersuite negotiation (see Section 3.2.7.). The extent of the security of the negotiation is commensurate with the security of the KDF used to derive the MICs, the length and entropy of the shared secret used by the KDF, and the length of the MICs.

5.16. Random Number Generation

EAP-SAKE supports key derivation from a 32-byte Root Secret. The entropy of all other keys derived from it is reduced somewhat through the use of keyed hash functions (e.g. KDF). Thus, assuming optimistically that the effective key strength of the Root Secret is 32 bytes, the effective key strengths of the derived keys is at most the effective key strength of the Root Secret quantities they are derived from: EMSK, at most 16 bytes; MSK, at most 16 bytes.

6. Security Claims

This section provides the security claims as required by [EAP].

[a] Mechanism: EAP-SAKE is a challenge/response authentication and key establishment mechanism based on a symmetric pre-shared secret.

[b] Security claims. EAP-SAKE provides:

Mutual authentication (Section 5.3)

Integrity protection (Section 5.4)

Replay protection (Section 5.5)

Confidentiality (optional, Section 5.6 and Section 5.13)

Key derivation (Section 5.7)

Dictionary attack protection (Section 5.8)

Protected result indication of successful authentication from Server and from Peer (Section 5.10)

Session independence (Section 5.12)

[c] Key strength. EAP-SAKE supports key derivation with 256-bit effective key strength (Section 5.7)

[d] Description of key hierarchy: see Section 3.2.5.

[e] Indication of vulnerabilities: EAP-Make does not provide:

Fast reconnect

Fragmentation

Channel binding

Cryptographic binding

7. Acknowledgements

Thanks to R. Dynarski for his helpful comments.

8. References

8.1. Normative References

[AES] National Institute of Standards and Technology, "Federal Information Processing Standards (FIPS) Publication 197, Advanced Encryption Standard (AES)", November 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[CBC] National Institute of Standards and Technology, NIST Special Publication 800-38A, "Recommendation for Block Cipher Modes of Operation - Methods and Techniques", December 2001. http://csrc.nist.gov/publications/drafts/Draft-NIST_SP800-38D_Public_Comment.pdf

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

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

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

[IEEE802.11i] "IEEE Standard for Information Technology-Telecommunications and Information Exchange between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements", June 2004.

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

[SHA1] National Institute of Standards and Technology, U.S. Department of Commerce, Federal Information Processing Standard (FIPS) Publication 180-1, "Secure Hash Standard", April 1995.

8.2. Informative References

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

[RFC4086] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

Authors' Addresses

Michaela Vanderveen
Qualcomm Flarion Technologies
135 Rte. 202/206 South
Bedminster, NJ 07921
USA

EMail: mvandervn@yahoo.com

Hesham Soliman
Qualcomm Flarion Technologies
135 Rte. 202/206 South
Bedminster, NJ 07921
USA

EMail: solimanhs@gmail.com

Full Copyright Statement

Intellectual Property

Acknowledgement

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