Network Working Group
Internet-Draft
Expires: October 4, 2004

J. Arkko
Ericsson
H. Haverinen
Nokia
April 5, 2004

Extensible Authentication Protocol Method for UMTS Authentication and Key Agreement (EAP-AKA)
draft-arkko-pppext-eap-aka-12.txt

Status of this Memo

Copyright Notice

Abstract

This document specifies an Extensible Authentication Protocol (EAP) mechanism for authentication and session key distribution using the Universal Mobile Telecommunications System (UMTS) Authentication and Key Agreement (AKA) mechanism. UMTS AKA is based on symmetric keys, and runs typically in a UMTS Subscriber Identity Module, a smart card like device.

EAP-AKA includes optional identity privacy support, optional result indications, and an optional fast re-authentication procedure.

Table of Contents

1. Introduction and Motivation 4
2. Terms and Conventions Used in This Document 5
3. Protocol Overview 8
4. Operation 12
4.1 Identity Management 13
4.1.1 Format, Generation and Usage of Peer Identities 13
4.1.2 Communicating the Peer Identity to the Server 19
4.1.3 Message Sequence Examples (Informative) 24
4.2 Fast Re-authentication 30
4.2.1 General 30
4.2.2 Comparison to UMTS AKA 31
4.2.3 Fast Re-authentication Identity 32
4.2.4 Fast Re-authentication Procedure 33
4.2.5 Fast Re-authentication Procedure when Counter is Too Small 35
4.3 EAP-AKA Notifications 37
4.3.1 General 37
4.3.2 Result Indications 38
4.4 Error Cases 39
4.4.1 Peer Operation 39
4.4.2 Server Operation 40
4.4.3 EAP-Failure 40
4.4.4 EAP-Success 41
4.5 Key Generation 42
5. Message Format and Protocol Extensibility 44
5.1 Message Format 44
5.2 Protocol Extensibility 45
6. Messages 46
6.1 EAP-Request/AKA-Identity 46
6.2 EAP-Response/AKA-Identity 46
6.3 EAP-Request/AKA-Challenge 47
6.4 EAP-Response/AKA-Challenge 47
6.5 EAP-Response/AKA-Authentication-Reject 48
6.6 EAP-Response/AKA-Synchronization-Failure 48
6.7 EAP-Request/AKA-Reauthentication 49
6.8 EAP-Response/AKA-Reauthentication 49
6.9 EAP-Response/AKA-Client-Error 50
6.10 EAP-Request/AKA-Notification 50
6.11 EAP-Response/AKA-Notification 50
7. Attributes 51

7.1 Table of Attributes 51
7.2 AT_PERMANENT_ID_REQ 52
7.3 AT_ANY_ID_REQ 52
7.4 AT_FULLAUTH_ID_REQ 53
7.5 AT_IDENTITY 53
7.6 AT_RAND 54

7.7 AT_AUTN 54
7.8 AT_RES 54
7.9 AT_AUTS 55
7.10 AT_NEXT_PSEUDONYM 55
7.11 AT_NEXT_REAUTH_ID 56
7.12 AT_IV, AT_ENCR_DATA and AT_PADDING 56
7.13 AT_CHECKCODE 58
7.14 AT_RESULT_IND 60
7.15 AT_MAC 61
7.16 AT_COUNTER 62
7.17 AT_COUNTER_TOO_SMALL 62
7.18 AT_NONCE_S 62
7.19 AT_NOTIFICATION 63
7.20 AT_CLIENT_ERROR_CODE 64
8. IANA and Protocol Numbering Considerations 64
9. Security Considerations 66
9.1 Identity Protection 66
9.2 Mutual Authentication 66
9.3 Flooding the Authentication Centre 66
9.4 Key Derivation 67
9.5 Brute-Force and Dictionary Attacks 67
9.6 Protection, Replay Protection and Confidentiality 67
9.7 Negotiation Attacks 68
9.8 Protected Result Indications 68
9.9 Man-in-the-middle Attacks 69
9.10 Generating Random Numbers 69
10. Security Claims 69
11. Acknowledgements and Contributions 70
Normative References 71
Informative References 72
Authors' Addresses 73
A. Pseudo-Random Number Generator 73
Intellectual Property and Copyright Statements 74

1. Introduction and Motivation

This document specifies an Extensible Authentication Protocol (EAP) mechanism for authentication and session key distribution using the UMTS AKA authentication mechanism [TS 33.102]. UMTS is a global third generation mobile network standard.

AKA is based on challenge-response mechanisms and symmetric cryptography. AKA typically runs in a UMTS Subscriber Identity Module (USIM). Compared to the GSM mechanism, UMTS AKA provides substantially longer key lengths and mutual authentication.

The introduction of AKA inside EAP allows several new applications. These include the following:

- The use of the AKA also as a secure PPP authentication method in devices that already contain an USIM.
- The use of the third generation mobile network authentication infrastructure in the context of wireless LANs
- Relying on AKA and the existing infrastructure in a seamless way with any other technology that can use EAP.

AKA works in the following manner:

- The USIM and the home environment have agreed on a secret key beforehand.
- The actual authentication process starts by having the home environment produce an authentication vector, based on the secret

key and a sequence number. The authentication vector contains a random part RAND, an authenticator part AUTN used for authenticating the network to the USIM, an expected result part XRES, a session key for integrity check IK, and a session key for encryption CK.

- The RAND and the AUTN are delivered to the USIM.
- The USIM verifies the AUTN, again based on the secret key and the sequence number. If this process is successful (the AUTN is valid and the sequence number used to generate AUTN is within the correct range), the USIM produces an authentication result, RES and sends this to the home environment.
- The home environment verifies the correct result from the USIM. If the result is correct, IK and CK can be used to protect further communications between the USIM and the home environment.

When verifying AUTN, the USIM may detect that the sequence number the network uses is not within the correct range. In this case, the USIM calculates a sequence number synchronization parameter AUTS and sends it to the network. AKA authentication may then be retried with a new authentication vector generated using the synchronized sequence

number.

For a specification of the AKA mechanisms and how the cryptographic values AUTN, RES, IK, CK and AUTS are calculated, see [TS 33.102].

In EAP-AKA, the EAP server node obtains the authentication vectors, compares RES and XRES, and uses CK and IK in key derivation.

In the third generation mobile networks, AKA is used both for radio network authentication and IP multimedia service authentication purposes. Different user identities and formats are used for these; the radio network uses the International Mobile Subscriber Identifier (IMSI), whereas the IP multimedia service uses the Network Access Identifier (NAI) [RFC2486].

2. Terms and Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The terms and abbreviations "authenticator", "backend authentication server", "EAP server", "peer", "Silently Discard", "Master Session Key (MSK)", and "Extended Master Session Key (EMSK)" in this document are to be interpreted as described in [EAP].

This document frequently uses the following terms and abbreviations:

AAA protocol

Authentication, Authorization and Accounting protocol

AKA

Authentication and Key Agreement

AuC

Authentication Centre. The mobile network element that can authenticate subscribers either in GSM or in UMTS networks.

EAP

Extensible Authentication Protocol
[EAP]

GSM

Global System for Mobile communications.

NAI

Network Access Identifier
[RFC2486]

AUTN

Authentication value generated by the AuC which together with the RAND authenticates the server to the peer, 128 bits
[TS 33.102]

AUTS

A value generated by the peer upon experiencing a synchronization failure, 112 bits.

Fast Re-authentication Identity

A fast re-authentication identity of the peer, including an NAI realm portion in environments where a realm is used. Used on re-authentication only.

Fast Re-authentication Username

The username portion of fast re-authentication identity, ie. not including any realm portions.

Nonce

A value that is used at most once or that is never repeated within the same cryptographic context. In general, a nonce can be predictable (e.g. a counter) or unpredictable (e.g. a random value). Since some cryptographic properties may depend on the randomness of the nonce, attention should be paid to whether a nonce is required to be random or not. In this document, the term nonce is only used to denote random nonces, and it is not used to denote counters.

Permanent Identity

The permanent identity of the peer, including an NAI realm portion in environments where a realm is used. The permanent identity is usually based on the IMSI. Used on full authentication only.

Permanent Username

The username portion of permanent identity, ie. not including any realm portions.

Pseudonym Identity

A pseudonym identity of the peer, including an NAI realm portion in environments where a realm is used. Used on full authentication

only.

Pseudonym Username

The username portion of pseudonym identity, ie. not including any realm portions.

RAND

Random number generated by the AuC, 128 bits
[TS 33.102]
.

RES

Authentication result from the peer, which together with the RAND authenticates the peer to the server, 128 bits
[TS 33.102]

SQN

Sequence number used in the authentication process, 48 bits
[TS 33.102]

SIM

Subscriber Identity Module. The SIM is traditionally a smart card distributed by a GSM operator.

SRES

The authentication result parameter in GSM, corresponds to the RES parameter in UMTS aka, 32 bits.

USIM

UMTS Subscriber Identity Module. USIM is an application that is resident e.g. on smart cards distributed by UMTS operators.

3. Protocol Overview

Figure 1 shows the basic successful full authentication exchange in EAP-AKA, when optional result indications are not used. The authenticator typically communicates with an EAP server that is located on a backend authentication server using an AAA protocol. The authenticator shown in the figure is often simply relaying EAP messages to and from the EAP server, but these back end AAA communications are not shown. At the minimum, EAP-AKA uses two roundtrips to authorize the user and generate session keys. As in other EAP schemes, an identity request/response message pair is usually exchanged first. On full authentication, the peer's identity response includes either the user's International Mobile Subscriber Identity (IMSI), or a temporary identity (pseudonym) if identity privacy is in effect, as specified in Section 4.1. (As specified in [EAP], the initial identity request is not required, and MAY be bypassed in cases where the network can presume the identity, such as when using leased lines, dedicated dial-ups, etc. Please see also Section 4.1.2 for specification how to obtain the identity via EAP AKA messages.)

After obtaining the subscriber identity, the EAP server obtains an authentication vector (RAND, AUTN, RES, CK, IK) for use in authenticating the subscriber. From the vector, the EAP server derives the keying material, as specified in Section 4.5. The vector may be obtained by contacting an Authentication Centre (AuC) on the UMTS network; per UMTS specifications, several vectors may be obtained at a time. Vectors may be stored in the EAP server for use at a later time, but they may not be reused.

Next, the EAP server starts the actual AKA protocol by sending an EAP-Request/AKA-Challenge message. EAP-AKA packets encapsulate parameters in attributes, encoded in a Type, Length, Value format. The packet format and the use of attributes are specified in Section 5. The EAP-Request/AKA-Challenge message contains a RAND random number (AT_RAND) and a network authentication token (AT_AUTN), and a message authentication code AT_MAC. The EAP-Request/AKA-Challenge message MAY optionally contain encrypted data, which is used for identity privacy and fast re-authentication support, as described in Section 4.1. The AT_MAC attribute contains a message authentication code covering the EAP packet. The encrypted data is not shown in the figures of this section.

The peer runs the AKA algorithm (typically using a USIM) and verifies the AUTN. If this is successful, the peer is talking to a legitimate EAP server and proceeds to send the EAP-Response/AKA-Challenge. This message contains a result parameter that allows the EAP server in turn to authenticate the peer, and the AT_MAC attribute to integrity

protect the EAP message.

The EAP server verifies that the RES and the MAC in the EAP-Response/AKA-Challenge packet are correct. Because protected success indications are not used in this example, the EAP server sends the EAP-Success packet, indicating that the authentication was successful. (Protected success indications are discussed in Section 4.3.2.) The EAP server may also include derived keying material in the message it sends to the authenticator. The peer has derived the same keying material, so the authenticator does not forward the keying material to the peer along with EAP-Success.

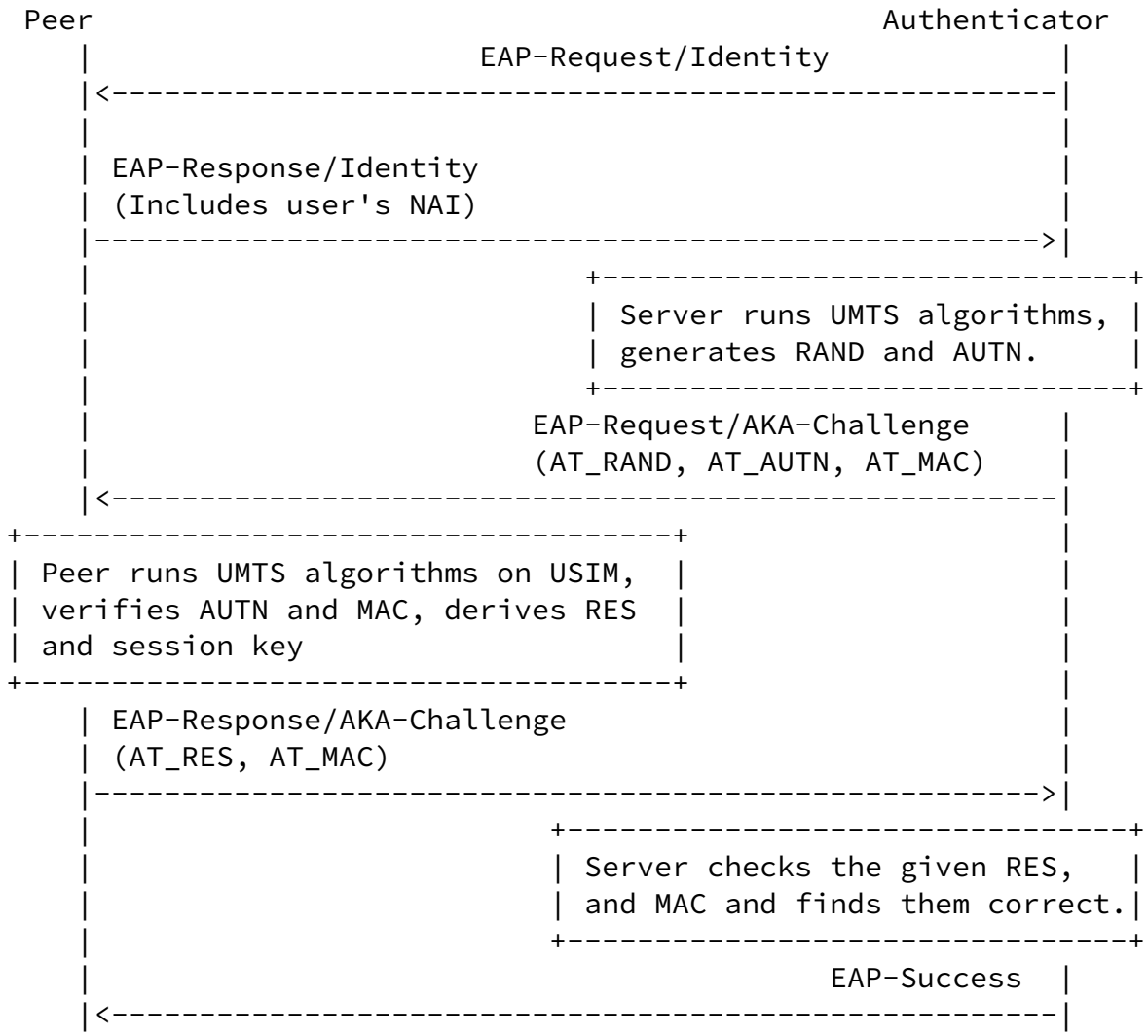


Figure 1: EAP-AKA full authentication procedure

Figure 2 shows how the EAP server rejects the Peer due to a failed authentication.

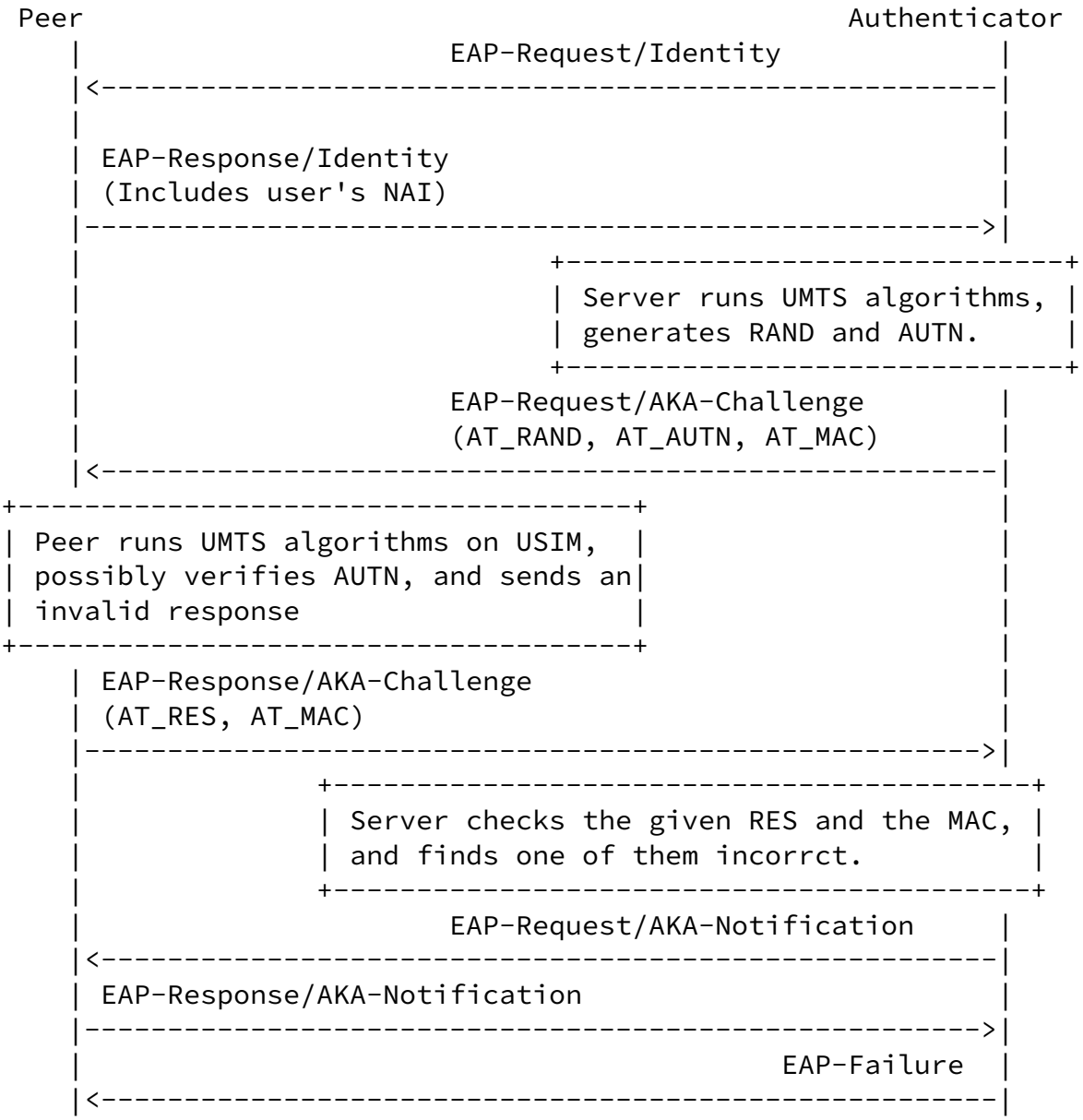


Figure 2: Peer authentication fails

Figure 3 shows the peer rejecting the AUTN of the EAP server.

The peer sends an explicit error message (EAP-Response/ AKA-Authentication-Reject) to the EAP server, as usual in AKA when AUTN is incorrect. This allows the EAP server to produce the same error statistics as AKA in general produces in UMTS.

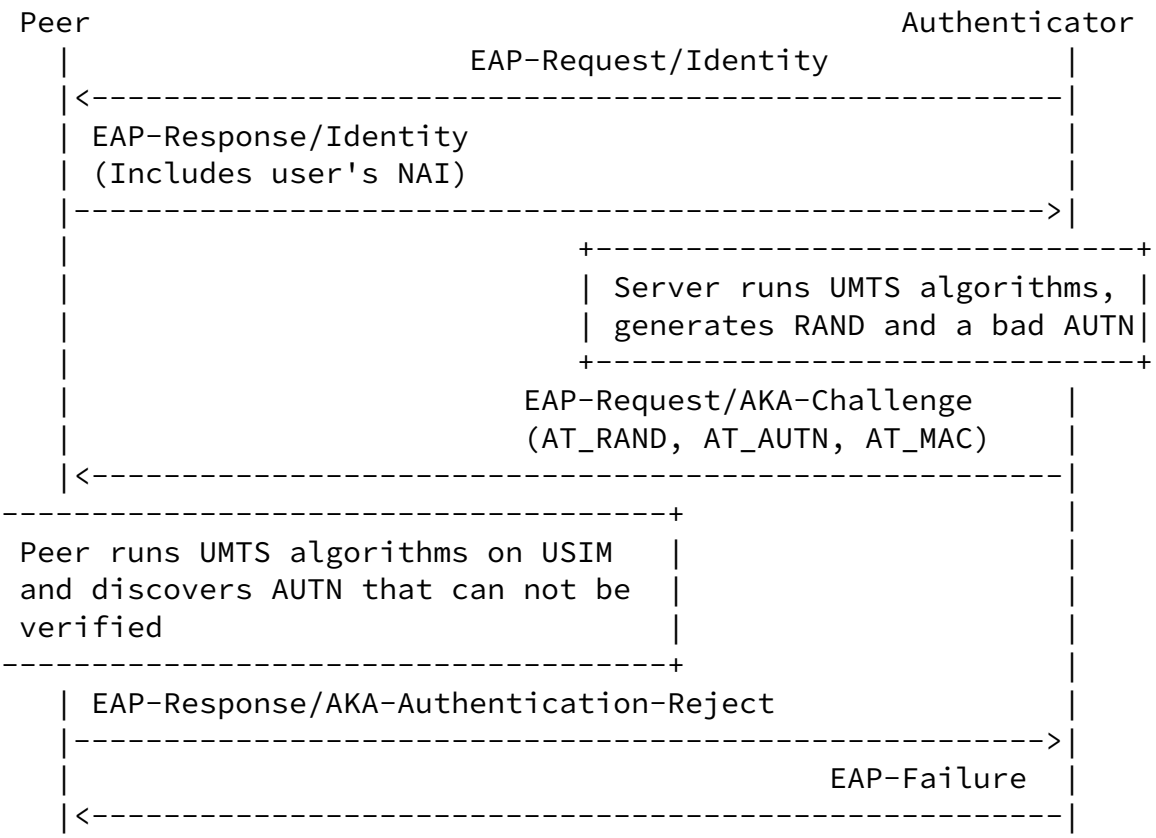
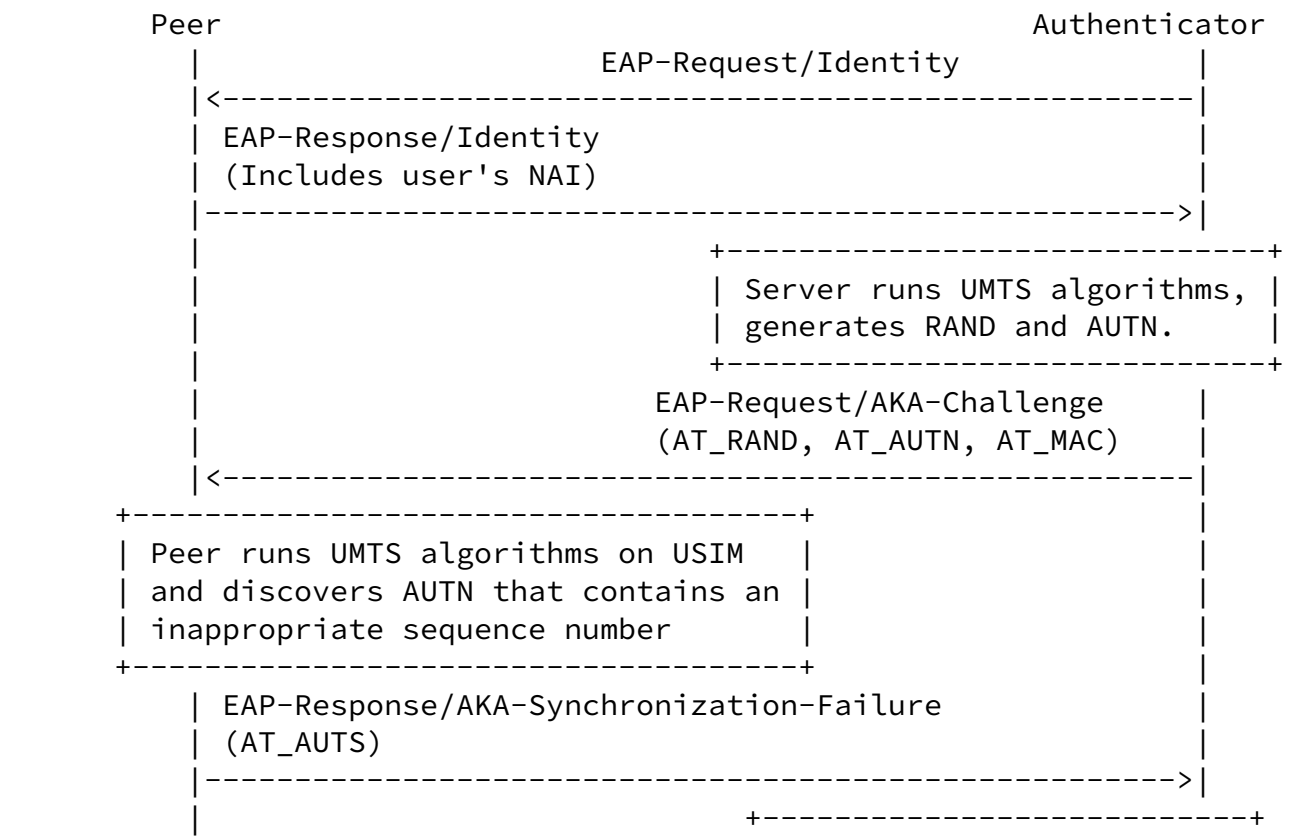


Figure 3: Network authentication fails

The AKA uses shared secrets between the Peer and the Peer's home operator together with a sequence number to actually perform an authentication. In certain circumstances it is possible for the sequence numbers to get out of sequence. Figure 4 shows what happens

then.


Peer
Authenticator
EAP-Request/Identity
EAP-Response/Identity
(Includes user's NAI)
Server runs UMTS algorithms,
generates RAND and AUTN.
EAP-Request/AKA-Challenge
(AT_RAND, AT_AUTN, AT_MAC)
Peer runs UMTS algorithms on USIM
and discovers AUTN that contains an
inappropriate sequence number
EAP-Response/AKA-Synchronization-Failure
(AT_AUTS)

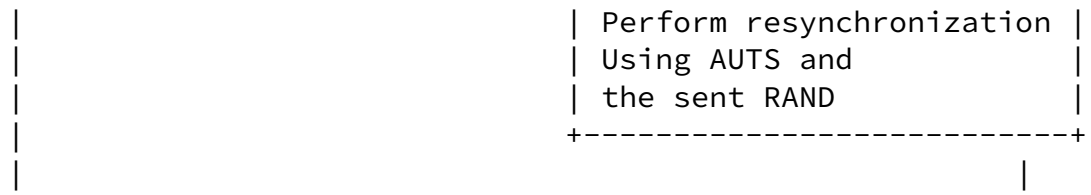


Figure 4: Sequence number synchronization

After the resynchronization process has taken place in the server and AAA side, the process continues by the server side sending a new EAP-Request/AKA-Challenge message.

In addition to the full authentication scenarios described above, EAP-AKA includes a fast re-authentication procedure, which is specified in Section 4.2. Fast re-authentication is based on keys derived on full authentication. If the peer has maintained state information for re- authentication and wants to use fast re-authentication, then the peer indicates this by using a specific fast re-authentication identity instead of the permanent identity or a pseudonym identity. The fast re-authentication procedure is described in Section 4.2.

4. Operation

4.1 Identity Management

4.1.1 Format, Generation and Usage of Peer Identities

4.1.1.1 General

In the beginning of EAP authentication, the Authenticator or the EAP server usually issues the EAP-Request/Identity packet to the peer. The peer responds with EAP-Response/Identity, which contains the user's identity. The formats of these packets are specified in [EAP].

UMTS subscribers are identified with the International Mobile Subscriber Identity (IMSI) [TS 23.003]. The IMSI is composed of a three digit Mobile Country Code (MCC), a two or three digit Mobile Network Code (MNC) and a not more than 10 digit Mobile Subscriber

Identification Number (MSIN). In other words, the IMSI is a string of not more than 15 digits. MCC and MNC uniquely identify the GSM operator and help identify the AuC from which the authentication vectors need to be retrieved for this subscriber.

Internet AAA protocols identify users with the Network Access Identifier (NAI) [RFC2486]. When used in a roaming environment, the NAI is composed of a username and a realm, separated with "@" (username@realm). The username portion identifies the subscriber within the realm.

This section specifies the peer identity format used in EAP-AKA. In this document, the term identity or peer identity refers to the whole identity string that is used to identify the peer. The peer identity may include a realm portion. "Username" refers to the portion of the peer identity that identifies the user, i.e. the username does not include the realm portion.

4.1.1.2 Identity Privacy Support

EAP-AKA includes optional identity privacy (anonymity) support that can be used to hide the cleartext permanent identity and thereby to make the subscriber's EAP exchanges untraceable to eavesdroppers. Because the permanent identity never changes, revealing it would help observers to track the user. The permanent identity is usually based on the IMSI, which may further help the tracking, because the same identifier may be used in other contexts as well. Identity privacy is based on temporary identities, or pseudonyms, which are equivalent to but separate from the Temporary Mobile Subscriber Identities (TMSI) that are used on cellular networks. Please see Section 9.1 for security considerations regarding identity privacy.

4.1.1.3 Username Types in EAP-AKA Identities

There are three types of usernames in EAP-AKA peer identities:

(1) Permanent usernames. For example, 0123456789098765@myoperator.com might be a valid permanent identity. In this example, 0123456789098765 is the permanent username.

(2) Pseudonym usernames. For example, 2s7ah6n9q@myoperator.com might be a valid pseudonym identity. In this example, 2s7ah6n9q is the pseudonym username.

(3) Fast re-authentication usernames. For example, 43953754@myoperator.com might be a valid fast re-authentication identity. In this case, 43953754 is the fast re-authentication username.

The first two types of identities are only used on full authentication and the last one only on fast re-authentication. When the optional identity privacy support is not used, the non-pseudonym permanent identity is used on full authentication. The fast re-authentication exchange is specified in Section 4.2.

4.1.1.4 Username Decoration

In some environments, the peer may need to decorate the identity by prepending or appending the username with a string, in order to indicate supplementary AAA routing information in addition to the NAI realm. (The usage of a NAI realm portion is not considered to be decoration.) Username decoration is out of the scope of this document. However, it should be noted that username decoration might prevent the server from recognizing a valid username. Hence, although the peer MAY use username decoration in the identities the peer includes in EAP-Response/Identity, and the EAP server MAY accept a decorated peer username in this message, the peer or the EAP server MUST NOT decorate any other peer identities that are used in various EAP-AKA attributes. Only the identity used in EAP-Response/Identity may be decorated.

4.1.1.5 NAI Realm Portion

The peer MAY include a realm portion in the peer identity, as per the NAI format. The use of a realm portion is not mandatory.

If a realm is used, the realm MAY be chosen by the subscriber's home operator and it MAY a configurable parameter in the EAP-SIM peer implementation. In this case, the peer is typically configured with the NAI realm of the home operator. Operators MAY reserve a specific

realm name for EAP-AKA users. This convention makes it easy to

recognize that the NAI identifies a UMTS subscriber. Such reserved NAI realm may be useful as a hint as to the first authentication method to use during method negotiation. When the peer is using a pseudonym username instead of the permanent username, the peer selects the realm name portion similarly as it select the realm portion when using the permanent username.

If no configured realm name is available, the peer MAY derive the realm name from the MCC and MNC portions of the IMSI. A RECOMMENDED way to derive the realm from the IMSI using the realm 3gppnetwork.org will be specified in [Draft 3GPP TS 23.003].

Some old implementations derive the realm name from the IMSI by concatenating "mnc", the MNC digits of IMSI, ".mcc", the MCC digits of IMSI and ".owlan.org". For example, if the IMSI is 123456789098765, and the MNC is three digits long, then the derived realm name is "mnc456.mcc123.owlan.org". As there are no DNS servers running at owlan.org, these realm names can only be used with manually configured AAA routing. New implementations SHOULD use the mechanism specified in [Draft 3GPP TS 23.003] instead of owlan.org as soon as the 3GPP specification is finalized.

The IMSI is a string of digits without any explicit structure, so the peer may not be able to determine the length of the MNC portion. If the peer is not able to determine whether the MNC is two or three digits long, the peer MAY use a 3-digit MNC. If the correct length of the MNC is two, then the MNC used in the realm name includes the first digit of MSIN. Hence, when configuring AAA networks for operators that have 2-digit MNC's, the network SHOULD also be prepared for realm names with incorrect 3-digit MNC's.

4.1.1.6 Format of the Permanent Username

The non-pseudonym permanent username SHOULD be derived from the IMSI. In this case, the permanent username MUST be of the format "0" | IMSI, where the character "|" denotes concatenation. In other words, the first character of the username is the digit zero (ASCII value 30 hexadecimal), followed by the IMSI. The IMSI is an ASCII string that consists of not more than 15 decimal digits (ASCII values between 30 and 39 hexadecimal), one character per IMSI digit, in the order as specified in [TS 23.003]. For example, a permanent username derived from the IMSI 295023820005424 would be encoded as the ASCII string "0295023820005424" (byte values in hexadecimal notation: 30 32 39 35 30 32 33 38 32 30 30 30 35 34 32 34)

The EAP server MAY use the leading "0" as a hint to try EAP-AKA as the first authentication method during method negotiation, rather

than for example EAP-SIM. The EAP-AKA server MAY propose EAP-AKA even if the leading character was not "0".

Alternatively, an implementation MAY choose a permanent username that is not based on the IMSI. In this case the selection of the username, its format, and its processing is out of the scope of this document. In this case, the peer implementation MUST NOT prepend any leading characters to the username.

4.1.1.7 Generating Pseudonyms and Fast Re-authentication Identities by the Server

Pseudonym usernames and fast re-authentication identities are generated by the EAP server. The EAP server produces pseudonym usernames and fast re-authentication identities in an implementation-dependent manner. Only the EAP server needs to be able to map the pseudonym username to the permanent identity, or to recognize a fast re-authentication identity.

EAP-AKA 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 issuing a fast re-authentication identity, the EAP server may include a realm name in the identity to make the fast re-authentication request be forwarded to the same EAP server.

When generating fast re-authentication identities, the server SHOULD choose a fresh new fast re-authentication identity that is different from the previous ones used within a same reauthentication context. The fast re-authentication identity SHOULD include a random component. The random component works as a full authentication context identifier. A context-specific fast re-authentication identity can help the server to detect whether its fast re-authentication state information matches the peer's fast re-authentication state information (in other words whether the state information is from the same full authentication exchange). The random component also makes the fast re-authentication identities unpredictable, so an attacker cannot initiate a fast re-authentication exchange to get the server's EAP-Request/SIM/

Re-authentication packet.

Regardless of construction method, the pseudonym username MUST conform to the grammar specified for the username portion of an NAI. The fast re-authentication identity also MUST conform to the NAI grammar. The EAP servers that the subscribers of an operator can use MUST ensure that the pseudonym usernames and the username portions used in fast re-authentication identities they generate are unique.

In any case, it is necessary that permanent usernames, pseudonym usernames and fast re-authentication usernames are separate and recognizable from each other. It is also desirable that EAP-SIM and EAP-AKA user names be recognizable from each other as an aid for the server to which method to offer.

In general, it is the task of the EAP server and the policies of its administrator to ensure sufficient separation in the usernames. Pseudonym usernames and fast re-authentication usernames are both produced and used by the EAP server. The EAP server MUST compose pseudonym usernames and fast re-authentication usernames so that it can recognize if a NAI username is an EAP-AKA pseudonym username or an EAP-AKA fast re-authentication username. For instance, when the usernames have been derived from the IMSI, the server could use different leading characters in the pseudonym usernames and fast re-authentication usernames (e.g. the pseudonym could begin with a leading "2" character). When mapping a fast re-authentication identity to a permanent identity, the server SHOULD only examine the username portion of the fast re-authentication identity and ignore the realm portion of the identity.

Because the peer may fail to save a pseudonym username sent to in an EAP-Request/AKA-Challenge, for example due to malfunction, the EAP server SHOULD maintain at least the most recently used pseudonym username in addition to the most recently issued pseudonym username. If the authentication exchange is not completed successfully, then the server SHOULD NOT overwrite the pseudonym username that was issued during the most recent successful authentication exchange.

4.1.1.8 Transmitting Pseudonyms and Fast Re-authentication Identities to the Peer

The server transmits pseudonym usernames and fast re-authentication identities to the peer in cipher, using the AT_ENCR_DATA attribute.

The EAP-Request/AKA-Challenge message MAY include an encrypted pseudonym username and/or an encrypted fast re-authentication identity in the value field of the AT_ENCR_DATA attribute. Because identity privacy support and fast re-authentication are optional to implement, the peer MAY ignore the AT_ENCR_DATA attribute and always use the permanent identity. On fast re-authentication (discussed in

Section 4.2), the server MAY include a new encrypted fast re-authentication identity in the EAP-Request/AKA-Reauthentication message.

On receipt of the EAP-Request/AKA-Challenge, the peer MAY decrypt the encrypted data in AT_ENCR_DATA and if a pseudonym username is included, the peer may use the obtained pseudonym username on the next full authentication. If a fast re-authentication identity is included, then the peer MAY save it together with other fast re-authentication state information, as discussed in Section 4.2, for the next re- authentication.

If the peer does not receive a new pseudonym username in the EAP-Request/AKA-Challenge message, the peer MAY use an old pseudonym username instead of the permanent username on next full authentication. The username portions of fast re-authentication identities are one-time usernames, which the peer MUST NOT re-use. When the peer uses a fast re-authentication identity in an EAP exchange, the peer MUST discard the fast re-authentication identity and not re-use it in another EAP authentication exchange, even if the authentication exchange was not completed.

4.1.1.9 Usage of the Pseudonym by the Peer

When the optional identity privacy support is used on full authentication, the peer MAY use a pseudonym username received as part of a previous full authentication sequence as the username portion of the NAI. The peer MUST NOT modify the pseudonym username received in AT_NEXT_PSEUDONYM. However, as discussed above, the peer MAY need to decorate the username in some environments by appending or prepending the username with a string that indicates supplementary AAA routing information.

When using a pseudonym username in an environment where a realm portion is used, the peer concatenates the received pseudonym username with the "@" character and a NAI realm portion. The selection of the NAI realm is discussed above. The peer can select the realm portion similarly regardless of whether it uses the permanent username or a pseudonym username.

4.1.1.10 Usage of the Fast Re-authentication Identity by the Peer

On fast re-authentication, the peer uses the fast re-authentication identity, received as part of the previous authentication sequence. A new fast re-authentication identity may be delivered as part of both full authentication and fast re-authentication. The peer MUST NOT modify the username part of the fast re-authentication identity received in AT_NEXT_REAUTH_ID, except in cases when username

decoration is required. Even in these cases, the "root" fast re-authentication username must not be modified, but it may be appended or prepended with another string.

4.1.2 Communicating the Peer Identity to the Server

4.1.2.1 General

The peer identity MAY be communicated to the server with the EAP-Response/Identity message. This message MAY contain the permanent identity, a pseudonym identity, or a fast re-authentication identity. If the peer uses the permanent identity or a pseudonym identity, which the server is able to map to the permanent identity, then the authentication proceeds as discussed in the overview of Section 3. If the peer uses a fast re-authentication identity, and if the fast re-authentication identity matches with a valid fast re-authentication identity maintained by the server , then a fast re-authentication exchange is performed, as described in Section 4.2.

The peer identity can also be transmitted from the peer to the server using EAP-AKA messages instead of EAP-Response/Identity. In this case, the server includes an identity requesting attribute (AT_ANY_ID_REQ, AT_FULLAUTH_ID_REQ or AT_PERMANENT_ID_REQ) in the EAP-Request/AKA-Identity message, and the peer includes the AT_IDENTITY attribute, which contains the peer's identity, in the

EAP-Response/AKA-Identity message. The AT_ANY_ID_REQ attribute is a general identity requesting attribute, which the server uses if it does not specify which kind of an identity the peer should return in AT_IDENTITY. The server uses the AT_FULLAUTH_ID_REQ attribute to request either the permanent identity or a pseudonym identity. The server uses the AT_PERMANENT_ID_REQ attribute to request the peer to send its permanent identity. The EAP-Request/AKA-Challenge, EAP-Response/AKA-Challenge, or the packets used on fast re-authentication may optionally include the AT_CHECKCODE attribute, which enables the protocol peers to ensure the integrity of the AKA-Identity packets. AT_CHECKCODE is specified in Section 7.13.

The identity format in the AT_IDENTITY attribute is the same as in the EAP-Response/Identity packet (except that identity decoration is not allowed). The AT_IDENTITY attribute contains a permanent identity, a pseudonym identity or a fast re-authentication identity.

Obtaining the subscriber identity via EAP-AKA messages is useful if the server does not have any EAP-AKA peer identity at the beginning of the EAP-AKA exchange or does not recognize the identity the peer used in EAP-Response/Identity. This may happen if, for example, the EAP-Response/Identity has been issued by some EAP method other than EAP-AKA or if intermediate entities or software layers in the peer

have modified the identity string in the EAP-Response/Identity packet. Also, some EAP layer implementations may cache the identity string from the first EAP authentication and do not obtain a new identity string from the EAP method implementation on subsequent authentication exchanges.

As the identity string is used in key derivation, any of these cases will result in failed authentication unless the EAP server uses EAP-AKA attributes to obtain an unmodified copy of the identity string. Therefore, unless the EAP server can be certain that no intermediate element or software layer has modified the EAP-Response/Identity packet, the EAP server MUST use the EAP-AKA attributes to obtain the identity, even if the identity received in EAP-Response/Identity was valid.

Please note that the EAP-AKA peer and the EAP-AKA server only process the AT_IDENTITY attribute and entities that only pass through EAP packets do not process this attribute. Hence, if the EAP server is

not co-located in the authenticator, then the authenticator and other intermediate AAA elements (such as possible AAA proxy servers) will continue to refer to the peer with the original identity from the EAP-Response/Identity packet regardless of whether the AT_IDENTITY attribute is used in EAP-AKA to transmit another identity.

4.1.2.2 Choice of Identity for the EAP-Response/Identity

If EAP-AKA peer is started upon receiving an EAP-Request/Identity message, then the peer performs the following steps.

If the peer has maintained fast re-authentication state information and if the peer wants to use fast re-authentication, then the peer transmits the fast re-authentication identity in EAP-Response/ Identity.

Else, if the peer has a pseudonym username available, then the peer transmits the pseudonym identity in EAP-Response/Identity.

In other cases, the peer transmits the permanent identity in EAP-Response/Identity.

4.1.2.3 Server Operation in the Beginning of EAP-AKA Exchange

If the EAP server has not received any EAP-AKA peer identity (permanent identity, pseudonym identity or fast re-authentication identity) from the peer when sending the first EAP-AKA request, or if the EAP server has received an EAP-Response/Identity packet but the contents do not appear to be a valid permanent identity, pseudonym identity or a re- authentication identity, then the server MUST

request an identity from the peer using one of the methods below.

The server sends the EAP-Request/AKA-Identity message with the AT_PERMANENT_ID_REQ message to indicate that the server wants the peer to include the permanent identity in the AT_IDENTITY attribute of the EAP-Response/AKA-Identity message. This is done in the following cases:

- The server does not support fast re-authentication or identity privacy.
- The server received an identity that it recognizes as a pseudonym

identity but the server is not able to map the pseudonym identity to a permanent identity.

- The server issues the EAP-Request/AKA-Identity packet with the AT_FULLAUTH_ID_REQ attribute to indicate that the server wants the peer to include a full authentication identity (pseudonym identity or permanent identity) in the AT_IDENTITY attribute of the EAP-Response/AKA-Identity message. This is done in the following cases:
- The server does not support fast re-authentication and the server supports identity privacy
- The server received an identity that it recognizes as a re-authentication identity but the server is not able to map the re-authentication identity to a permanent identity

The server issues the EAP-Request/AKA-Identity packet with the AT_ANY_ID_REQ attribute to indicate that the server wants the peer to include an identity in the AT_IDENTITY attribute of the EAP-Response/SIM/Start message, and the server does not indicate any preferred type for the identity. This is done in other cases, such as when the server does not have any identity, or the server does not recognize the format of a received identity.

4.1.2.4 Processing of EAP-Request/AKA-Identity by the Peer

Upon receipt of an EAP-Request/AKA-Identity message, the peer MUST perform the following steps.

If the EAP-Request/AKA-Identity includes AT_PERMANENT_ID_REQ, and if the peer does not have a pseudonym available, then the peer MUST respond with EAP-Response/AKA-Identity and include the permanent identity in AT_IDENTITY. If the peer has a pseudonym available, then the peer MAY refuse to send the permanent identity; hence in this case the peer MUST either respond with EAP-Response/AKA-Identity and include the permanent identity in AT_IDENTITY or respond with EAP-Response/AKA-Client-Error packet with code "unable to process packet".

If the EAP-Request/AKA-Identity includes AT_FULL_AUTH_ID_REQ, and if the peer has a pseudonym available, then the peer SHOULD respond with EAP-Response/AKA-Identity and include the pseudonym identity in AT_IDENTITY. If the peer does not have a pseudonym when it receives

this message, then the peer MUST respond with EAP-Response/ AKA-Identity and include the permanent identity in AT_IDENTITY. The Peer MUST NOT use a fast re-authentication identity in the AT_IDENTITY attribute.

If the EAP-Request/AKA-Identity includes AT_ANY_ID_REQ, and if the peer has maintained fast re-authentication state information and the peer wants to use fast re-authentication, then the peer responds with EAP- Response/AKA-Identity and includes the fast re-authentication identity in AT_IDENTITY. Else, if the peer has a pseudonym identity available, then the peer responds with EAP-Response/AKA-Identity and includes the pseudonym identity in AT_IDENTITY. Else, the peer responds with EAP-Response/AKA-Identity and includes the permanent identity in AT_IDENTITY.

An EAP-AKA exchange may include several EAP/AKA-Identity rounds. The server may issue a second EAP-Request/AKA-Identity, if it was not able to recognize the identity the peer used in the previous AT_IDENTITY attribute. At most three EAP/AKA-Identity rounds can be used, so the peer MUST NOT respond to more than three EAP-Request/ AKA-Identity messages within an EAP exchange. The peer MUST verify that the sequence of EAP-Request/AKA-Identity packets the peer receives comply with the sequencing rules defined in this document. That is, AT_ANY_ID_REQ can only be used in the first EAP-Request/ AKA-Identity, in other words AT_ANY_ID_REQ MUST NOT be used in the second or third EAP-Request/AKA-Identity. AT_FULLAUTH_ID_REQ MUST NOT be used if the previous EAP-Request/AKA-Identity included AT_PERMANENT_ID_REQ. The peer operation in cases when it receives an unexpected attribute or an unexpected message is specified in Section 4.4.1.

4.1.2.5 Attacks against Identity Privacy

The section above specifies two possible ways the peer can operate upon receipt of AT_PERMANENT_ID_REQ. This is because a received AT_PERMANENT_ID_REQ does not necessarily originate from the valid network, but an active attacker may transmit an EAP-Request/ AKA-Identity packet with an AT_PERMANENT_ID_REQ attribute to the peer, in an effort to find out the true identity of the user. If the peer does not want to reveal its permanent identity, then the peer sends the EAP-Response/AKA-Client-Error packet with the error code "unable to process packet", and the authentication exchange terminates.

Basically, there are two different policies that the peer can employ with regard to AT_PERMANENT_ID_REQ. A "conservative" peer assumes that the network is able to maintain pseudonyms robustly. Therefore, if a conservative peer has a pseudonym username, the peer responds with EAP-Response/AKA-Client-Error to the EAP packet with AT_PERMANENT_ID_REQ, because the peer believes that the valid network is able to map the pseudonym identity to the peer's permanent identity. (Alternatively, the conservative peer may accept AT_PERMANENT_ID_REQ in certain circumstances, for example if the pseudonym was received a long time ago.) The benefit of this policy is that it protects the peer against active attacks on anonymity. On the other hand, a "liberal" peer always accepts the AT_PERMANENT_ID_REQ and responds with the permanent identity. The benefit of this policy is that it works even if the valid network sometimes loses pseudonyms and is not able to map them to the permanent identity.

4.1.2.6 Processing of AT_IDENTITY by the Server

When the server receives an EAP-Response/AKA-Identity message with the AT_IDENTITY (in response to the server's identity requesting attribute), the server MUST operate as follows.

If the server used AT_PERMANENT_ID_REQ, and if the AT_IDENTITY does not contain a valid permanent identity, then the server sends an EAP-Request/AKA-Notification packet with AT_NOTIFICATION code 16384 to terminate the EAP exchange. If the server recognizes the permanent identity and is able to continue, then the server proceeds with full authentication by sending EAP-Request/AKA-Challenge.

If the server used AT_FULLAUTH_ID_REQ, and if AT_IDENTITY contains a valid permanent identity or a pseudonym identity that the server can map to a valid permanent identity, then the server proceeds with full authentication by sending EAP-Request/AKA-Challenge. If AT_IDENTITY contains a pseudonym identity that the server is not able to map to a valid permanent identity, or an identity that the server is not able to recognize or classify, then the server sends EAP-Request/ AKA-Identity with AT_PERMANENT_ID_REQ.

If the server used AT_ANY_ID_REQ, and if the AT_IDENTITY contains a valid permanent identity or a pseudonym identity that the server can map to a valid permanent identity, then the server proceeds with full authentication by sending EAP-Request/ AKA-Challenge.

If the server used AT_ANY_ID_REQ, and if AT_IDENTITY contains a valid fast re-authentication identity and the server agrees on using re-authentication, then the server proceeds with fast re-authentication

by sending EAP-Request/AKA-Reauthentication (Section 4.2).

If the server used AT_ANY_ID_REQ, and if the peer sent an EAP-Response/AKA-Identity with AT_IDENTITY that contains an identity that the server recognizes as a fast re-authentication identity, but the server is not able to map the identity to a permanent identity, then the server sends EAP-Request/AKA-Identity with AT_FULLAUTH_ID_REQ.

If the server used AT_ANY_ID_REQ, and if AT_IDENTITY contains a valid fast re-authentication identity, which the server is able to map to a permanent identity, and if the server does not want to use fast re-authentication, then the server proceeds with full authentication by sending EAP-Request/AKA-Challenge.

If the server used AT_ANY_ID_REQ, and AT_IDENTITY contains an identity that the server recognizes as a pseudonym identity but the server is not able to map the pseudonym identity to a permanent identity, then the server sends EAP-Request/AKA-Identity with AT_PERMANENT_ID_REQ.

If the server used AT_ANY_ID_REQ, and AT_IDENTITY contains an identity that the server is not able to recognize or classify, then the server sends EAP-Request/AKA-Identity with AT_FULLAUTH_ID_REQ.

4.1.3 Message Sequence Examples (Informative)

This section contains non-normative message sequence examples to illustrate how the peer identity can be communicated to the server.

4.1.3.1 Usage of AT_ANY_ID_REQ

Obtaining the peer identity with EAP-AKA attributes is illustrated in Figure 5 below.

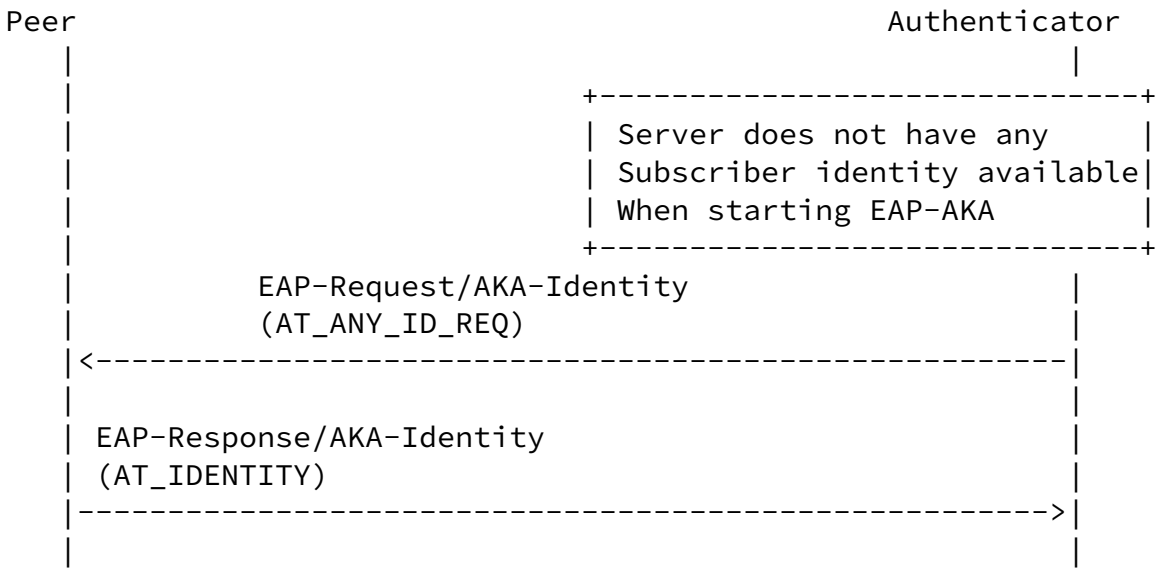


Figure 5: Usage of AT_ANY_ID_REQ

4.1.3.2 Fall Back on Full Authentication

Figure 6 illustrates the case when the server does not recognize the fast re-authentication identity the peer used in AT_IDENTITY.

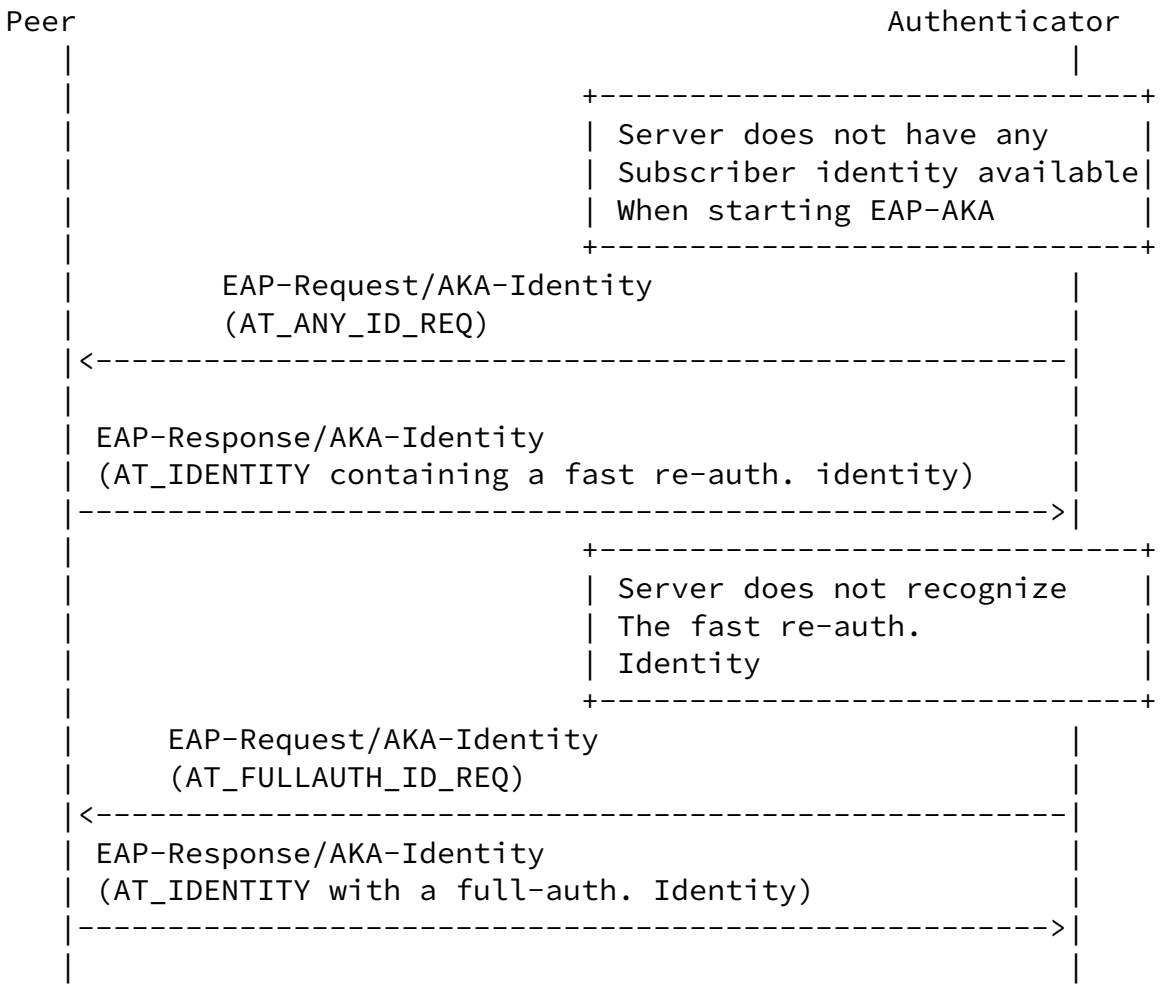

Peer
Authenticator
Server does not have any Subscriber identity available When starting EAP-AKA
EAP-Request/AKA-Identity
(AT_ANY_ID_REQ)
EAP-Response/AKA-Identity
(AT_IDENTITY containing a fast re-auth. identity)
Server does not recognize The fast re-auth. Identity
EAP-Request/AKA-Identity
(AT_FULLAUTH_ID_REQ)
EAP-Response/AKA-Identity
(AT_IDENTITY with a full-auth. Identity)

Figure 6: Fall back on full authentication

If the server recognizes the fast re-authentication identity, but still wants to fall back on full authentication, the server may issue the EAP-Request/AKA-Challenge packet. In this case, the full authentication procedure proceeds as usual.

4.1.3.3 Requesting the Permanent Identity 1

Figure 7 illustrates the case when the EAP server fails to decode a pseudonym identity included in the EAP-Response/Identity packet.

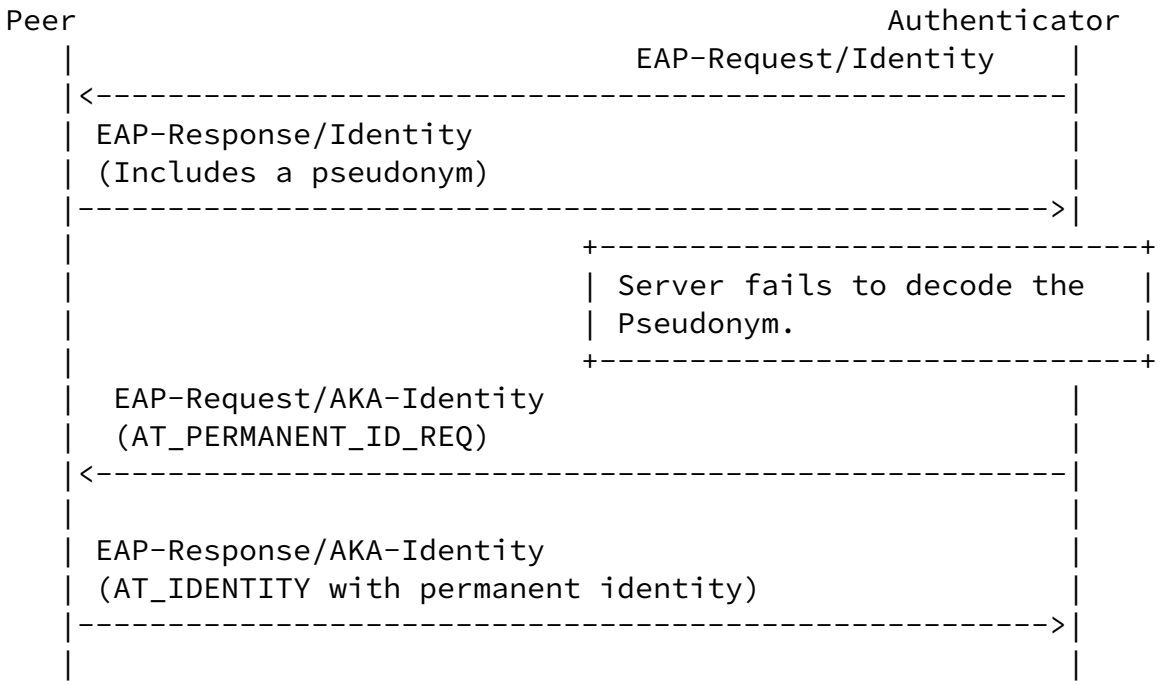
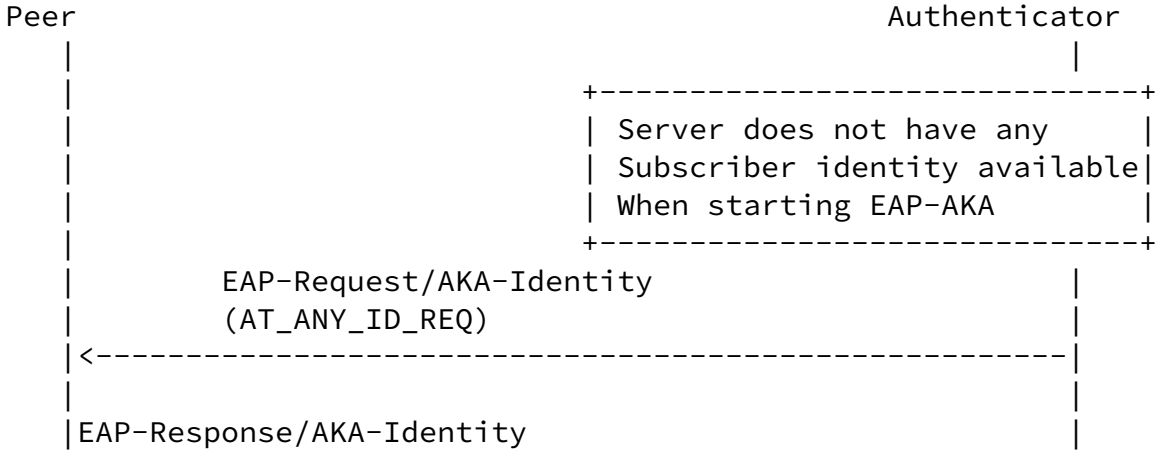


Figure 7: Requesting the permanent identity 1

If the server recognizes the permanent identity, then the authentication sequence proceeds as usual with the EAP Server issuing the EAP-Request/AKA-Challenge message.

4.1.3.4 Requesting the Permanent Identity 2

Figure 8 illustrates the case when the EAP server fails to decode the pseudonym included in the AT_IDENTITY attribute.



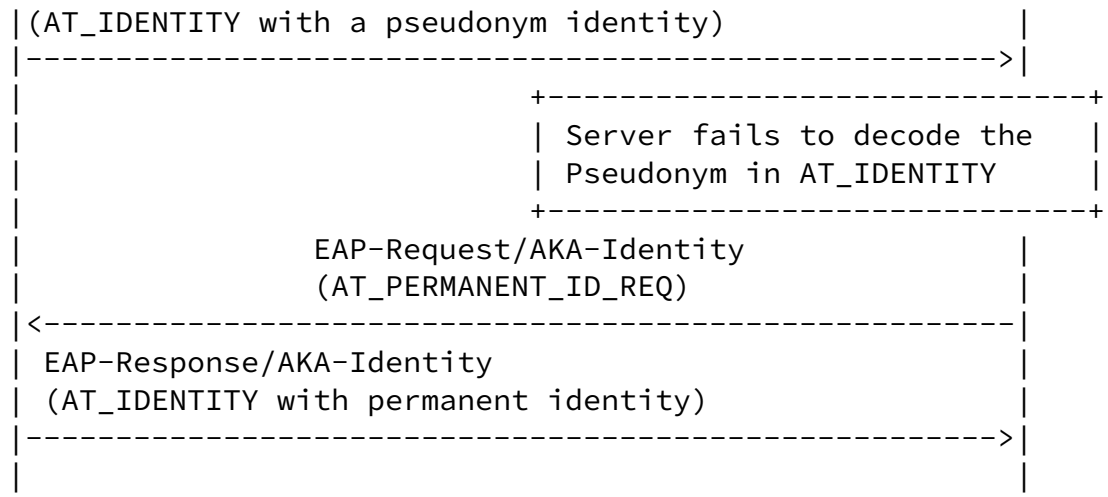
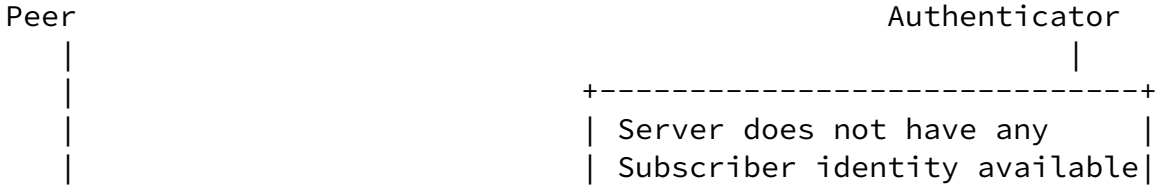
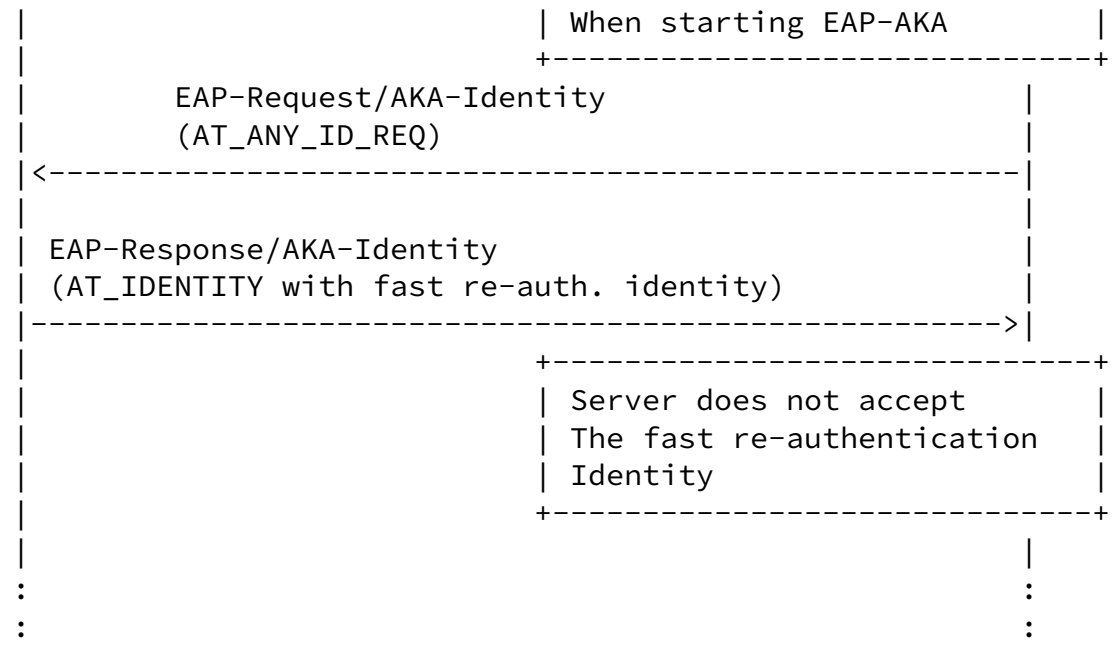


Figure 8: Requesting the permanent identity 2

4.1.3.5 Three EAP/AKA-Identity Round Trips

Figure 9 illustrates the case with three EAP/AKA-Identity round trips.




When starting EAP-AKA
EAP-Request/AKA-Identity
(AT_ANY_ID_REQ)
EAP-Response/AKA-Identity
(AT_IDENTITY with fast re-auth. identity)
Server does not accept
The fast re-authentication
Identity

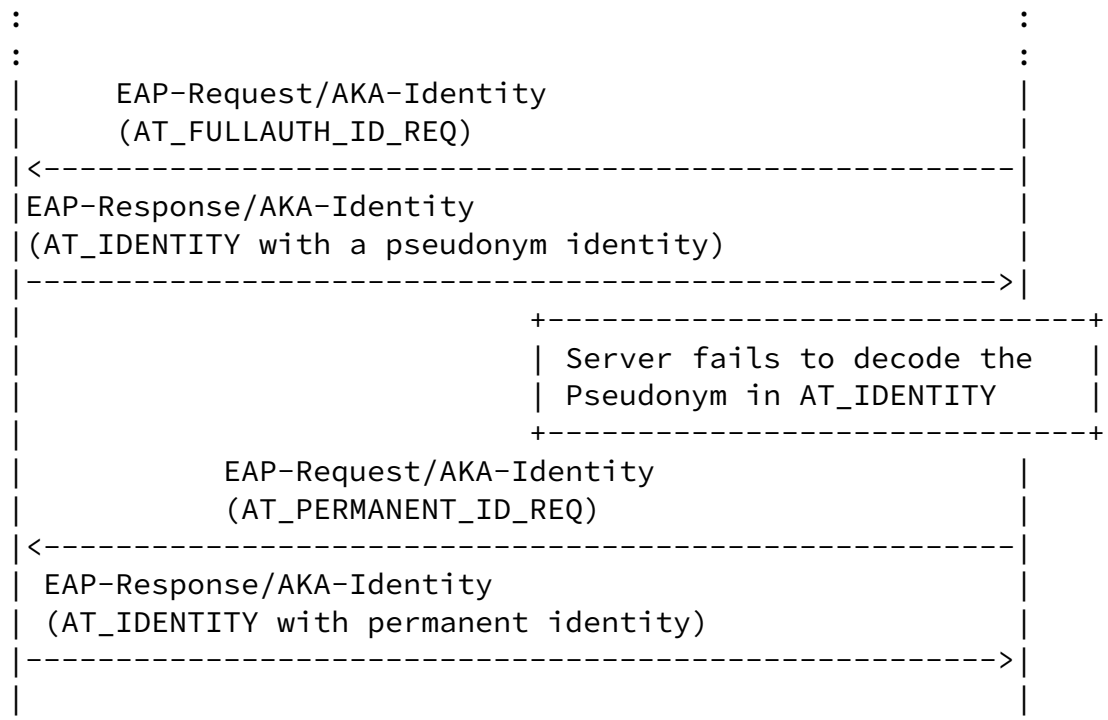


Figure 9: Three EAP-AKA Start rounds

After the last EAP-Response/AKA-Identity message, the full authentication sequence proceeds as usual.

4.2 Fast Re-authentication

4.2.1 General

In some environments, EAP authentication may be performed frequently. Because the EAP-AKA full authentication procedure makes use of the UMTS AKA algorithms, and it therefore requires fresh authentication vectors from the Authentication Centre, the full authentication procedure may result in many network operations when used very frequently. Therefore, EAP-AKA includes a more inexpensive fast re-authentication procedure that does not make use of the UMTS AKA algorithms and does not need new vectors from the Authentication Centre.

Fast re-authentication is optional to implement for both the EAP-AKA server and peer. On each EAP authentication, either one of the entities may also fall back on full authentication if they do not want to use fast re-authentication.

Fast re-authentication is based on the keys derived on the preceding full authentication. The same K_aut and K_encr keys as in full authentication are used to protect EAP-AKA packets and attributes, and the original Master Key from full authentication is used to

generate a fresh Master Session Key, as specified in Section 4.5.

The fast re-authentication exchange makes use of an unsigned 16-bit counter, included in the AT_COUNTER attribute. The counter has three goals: 1) it can be used to limit the number of successive reauthentication exchanges without full-authentication 2) it contributes to the keying material, and 3) it protects the peer and the server from replays. On full authentication, both the server and the peer initialize the counter to one. The counter value of at least one is used on the first fast re-authentication. On subsequent fast re-authentications, the counter MUST be greater than on any of the previous fast re-authentications. For example, on the second fast re-authentication, counter value is two or greater etc. The AT_COUNTER attribute is encrypted.

Both the peer and the EAP server maintain a copy of the counter. The EAP server sends its counter value to the peer in the fast re-authentication request. The peer MUST verify that its counter value is less than or equal to the value sent by the EAP server.

The server includes an encrypted server random nonce (AT_NONCE_S) in the fast re-authentication request. The AT_MAC attribute in the peer's response is calculated over NONCE_S to provide a challenge/ response authentication scheme. The NONCE_S also contributes to the new Master Session Key.

Both the peer and the server SHOULD have an upper limit for the number of subsequent fast re-authentications allowed before a full authentication needs to be performed. Because a 16-bit counter is used in fast re-authentication, the theoretical maximum number of re-authentications is reached when the counter value reaches FFFF hexadecimal. In order to use fast re-authentication, the peer and the EAP server need to store the following values: Master Key, latest counter value and the next fast re-authentication identity. K_aut, K_encr may either be stored or derived again from MK. The server may also need to store the permanent identity of the user.

4.2.2 Comparison to UMTS AKA

When analyzing the fast re-authentication exchange, it may be helpful to compare it with the UMTS Authentication and Key Agreement (AKA) exchange, which it resembles closely. The counter corresponds to the

UMTS AKA sequence number, NONCE_S corresponds to RAND, and AT_MAC in EAP-Request/AKA-Reauthentication corresponds to AUTN, the AT_MAC in EAP-Response/AKA-Reauthentication corresponds to RES, AT_COUNTER_TOO_SMALL corresponds to AUTS, and encrypting the counter corresponds to the usage of the Anonymity Key. Also the key generation on fast re-authentication with regard to random or fresh

material is similar to UMTS AKA -- the server generates the NONCE_S and counter values, and the peer only verifies that the counter value is fresh.

It should also be noted that encrypting the AT_NONCE_S, AT_COUNTER or AT_COUNTER_TOO_SMALL attributes is not important to the security of the fast re-authentication exchange.

4.2.3 Fast Re-authentication Identity

The fast re-authentication procedure makes use of separate re-authentication user identities. Pseudonyms and the permanent identity are reserved for full authentication only. If a fast re-authentication identity is lost and the network does not recognize it, the EAP server can fall back on full authentication. If the EAP server supports fast re-authentication, it MAY include the skippable AT_NEXT_REAUTH_ID attribute in the encrypted data of EAP- Request/ AKA-Challenge message. This attribute contains a new re-authentication identity for the next fast re-authentication. The attribute also works as a capability flag that indicates the fact that the server supports fast re-authentication, and that the server wants to continue using fast re-authentication within the current context. The peer MAY ignore this attribute, in which case it will use full authentication next time. If the peer wants to use fast re-authentication, it uses this fast re-authentication identity on next authentication. Even if the peer has a fast re-authentication identity, the peer MAY discard the re- authentication identity and use a pseudonym or the permanent identity instead, in which case full authentication MUST be performed. If the EAP server does not include the AT_NEXT_REAUTH_ID in the encrypted data of EAP-Request/ AKA-Challenge or EAP-Request/AKA-Reauthentication, then the peer MUST discard its current fast re-authentication state information and perform a full authentication next time.

In environments where a realm portion is needed in the peer identity,

the fast re-authentication identity received in AT_NEXT_REAUTH_ID MUST contain both a username portion and a realm portion, as per the NAI format. The EAP Server can choose an appropriate realm part in order to have the AAA infrastructure route subsequent fast re-authentication related requests to the same AAA server. For example, the realm part MAY include a portion that is specific to the AAA server. Hence, it is sufficient to store the context required for fast re-authentication in the AAA server that performed the full authentication.

The peer MAY use the fast re-authentication identity in the EAP-Response/Identity packet or, in response to server's AT_ANY_ID_REQ attribute, the peer MAY use the fast re-authentication

identity in the AT_IDENTITY attribute of the EAP-Response/ AKA-Identity packet. The peer MUST NOT modify the username portion of the fast re-authentication identity, but the peer MAY modify the realm portion or replace it with another realm portion.

Even if the peer uses a fast re-authentication identity, the server may want to fall back on full authentication, for example because the server does not recognize the fast re-authentication identity or does not want to use fast re-authentication. If the server was able to decode the fast re-authentication identity to the permanent identity, the server issues the EAP-Request/AKA-Challenge packet to initiate full authentication. If the server was not able to recover the peer's identity from the fast re-authentication identity, the server starts the full authentication procedure by issuing an EAP-Request/ AKA-Identity packet. This packet always starts a full authentication sequence if it does not include the AT_ANY_ID_REQ attribute.

4.2.4 Fast Re-authentication Procedure

Figure 10 illustrates the fast re-authentication procedure. In this example, the optional protected success indication is not used. Encrypted attributes are denoted with '*'. The peer uses its fast re-authentication identity in the EAP-Response/Identity packet. As discussed above, an alternative way to communicate the fast re-authentication identity to the server is for the peer to use the AT_IDENTITY attribute in the EAP-Response/AKA-Identity message. This latter case is not illustrated in the figure below, and it is only possible when the server requests the peer to send its identity by

including the AT_ANY_ID_REQ attribute in the EAP-Request/AKA-Identity packet.

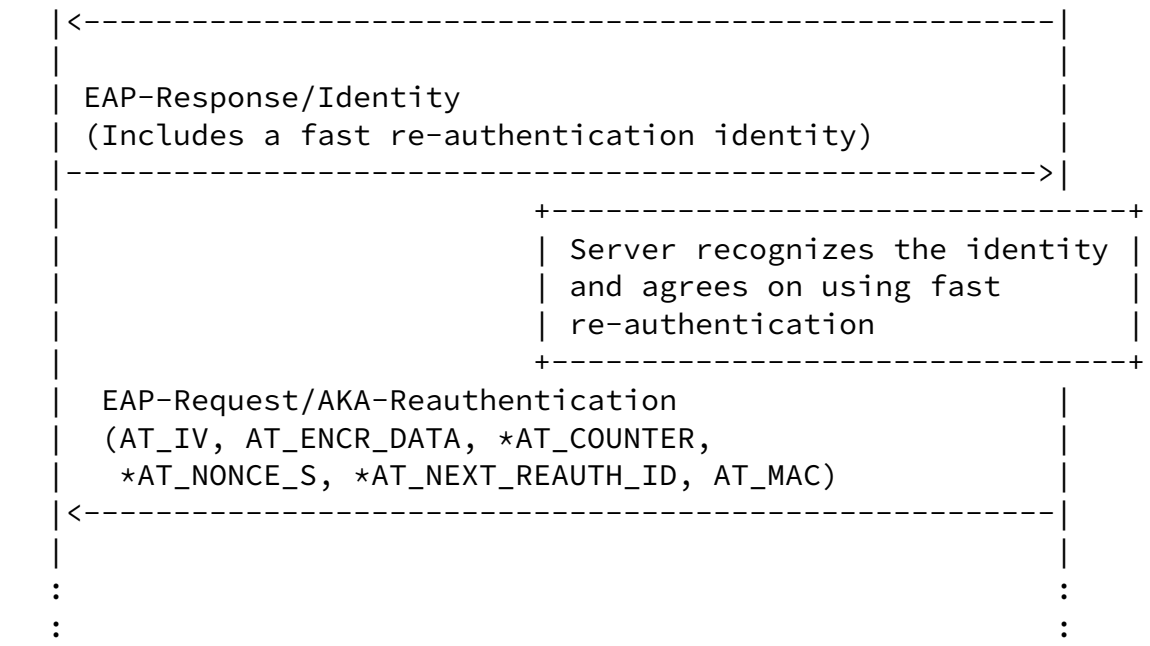
If the server recognizes the identity as a valid fast re-authentication identity, and if the server agrees on using fast re-authentication, then the server sends the EAP- Request/ AKA-Reauthentication packet to the peer. This packet MUST include the encrypted AT_COUNTER attribute, with a fresh counter value, the encrypted AT_NONCE_S attribute that contains a random number chosen by the server, the AT_ENCR_DATA and the AT_IV attributes used for encryption, and the AT_MAC attribute that contains a message authentication code over the packet. The packet MAY also include an encrypted AT_NEXT_REAUTH_ID attribute that contains the next fast re-authentication identity.

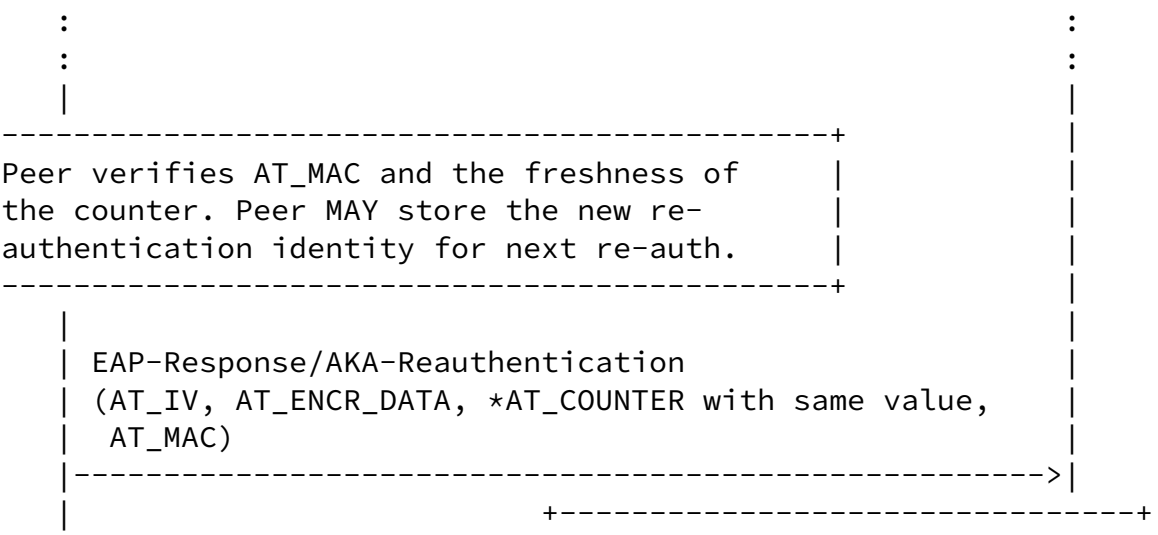
Fast re-authentication identities are one-time identities. If the peer does not receive a new fast re-authentication identity, it MUST use either the permanent identity or a pseudonym identity on the next authentication to initiate full authentication.

The peer verifies that AT_MAC is correct and that the counter value is fresh (greater than any previously used value). The peer MAY save the next fast re-authentication identity from the encrypted AT_NEXT_REAUTH_ID for next time. If all checks are successful, the peer responds with the EAP-Response/AKA-Reauthentication packet, including the AT_COUNTER attribute with the same counter value and the AT_MAC attribute.

The server verifies the AT_MAC attribute and also verifies that the counter value is the same that it used in the EAP-Request/AKA-Reauthentication packet. If these checks are successful, the fast re-authentication has succeeded and the server sends the EAP-Success packet to the peer.

If protected success indications (Section 4.3.2) were used, the EAP-Success packet would be preceded by an EAP-SIM notification round.


EAP-Response/Identity
(Includes a fast re-authentication identity)
Server recognizes the identity
and agrees on using fast
re-authentication
EAP-Request/AKA-Reauthentication
(AT_IV, AT_ENCR_DATA, *AT_COUNTER,
*AT_NONCE_S, *AT_NEXT_REAUTH_ID, AT_MAC)


Peer verifies AT_MAC and the freshness of
the counter. Peer MAY store the new re-
authentication identity for next re-auth.
EAP-Response/AKA-Reauthentication
(AT_IV, AT_ENCR_DATA, *AT_COUNTER with same value,
AT_MAC)

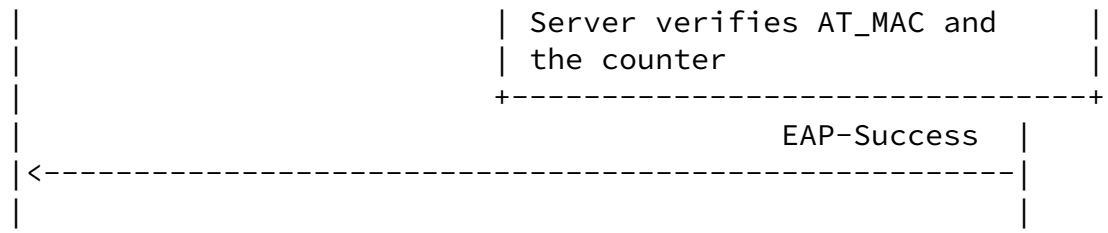
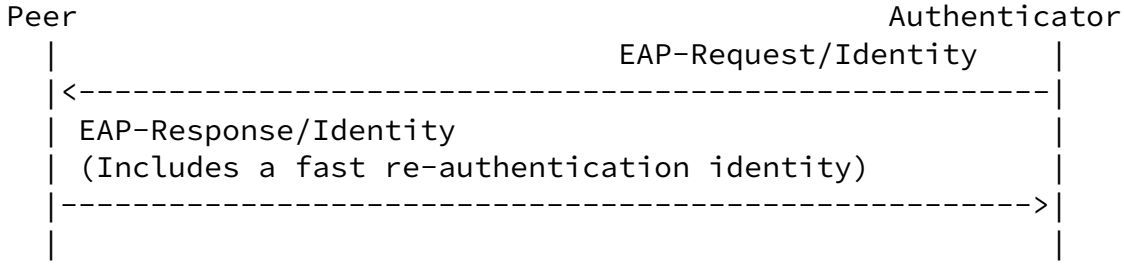


Figure 10: Reauthentication

4.2.5 Fast Re-authentication Procedure when Counter is Too Small

If the peer does not accept the counter value of EAP-Request/ AKA-Reauthentication, it indicates the counter synchronization problem by including the encrypted AT_COUNTER_TOO_SMALL in EAP-Response/AKA-Reauthentication. The server responds with EAP-Request/AKA-Challenge to initiate a normal full authentication procedure. This is illustrated in Figure 11. Encrypted attributes are denoted with '*'.



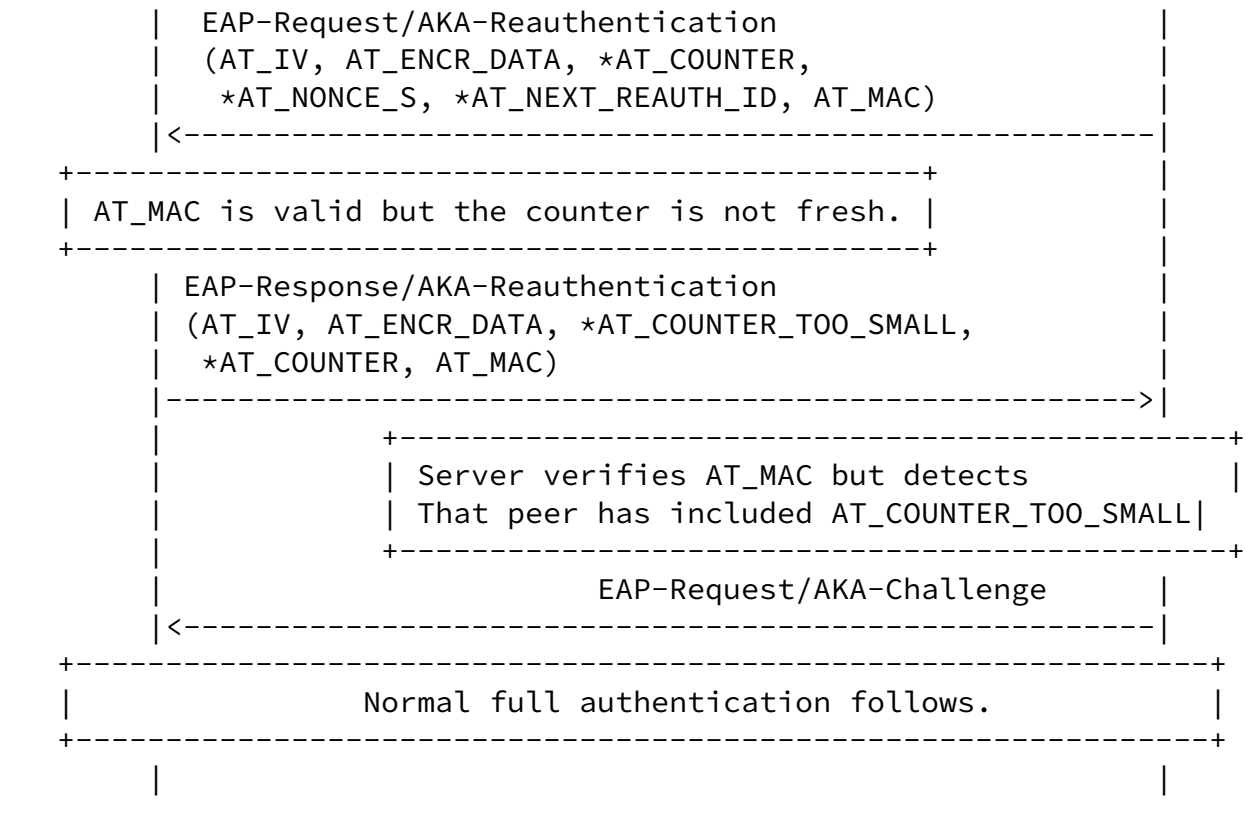


Figure 11: Fast re-authentication counter too small

In the figure above, the first three messages are similar to the basic fast re-authentication case. When the peer detects that the counter value is not fresh, it includes the AT_COUNTER_TOO_SMALL attribute in EAP-Response/AKA-Reauthentication. This attribute doesn't contain any data but it is a request for the server to initiate full authentication. In this case, the peer MUST ignore the contents of the server's AT_NEXT_REAUTH_ID attribute.

On receipt of AT_COUNTER_TOO_SMALL, the server verifies AT_MAC and verifies that AT_COUNTER contains the same counter value as in the EAP-Request/AKA-Reauthentication packet. If not, the server terminates the authentication exchange by sending the EAP-Request/ AKA-Notification packet with AT_NOTIFICATION code 16384. If all checks on the packet are successful, the server transmits a EAP-Request/AKA-Challenge packet and the full authentication procedure is performed as usual. Since the server already knows the subscriber identity, it MUST NOT use the EAP-Request/AKA-Identity

packet to request the identity.

4.3 EAP-AKA Notifications

4.3.1 General

The EAP server can use EAP-AKA notifications to convey localizable notifications and result indications (Section 4.3.2) to the peer.

The server MUST use notifications in cases discussed in Section 4.4.2. When the EAP server issues an EAP-Request/AKA-Notification packet to the peer, the peer MUST process the notification packet.The peer MAY show a notification message to the user and the peer MUST respond to the EAP server with an EAP-Response/AKA-Notification packet, even if the peer did not recognize the notification code.

An EAP-AKA full authentication exchange or a fast re-authentication exchange MUST NOT include more than one EAP-AKA notification round.

The notification code is a 16-bit number. The most significant bit is called the Failure bit (F bit). The F bit specifies whether the notification implies failure. The code values with the F bit set to zero (code values 0...32767) are used on unsuccessful cases. The receipt of a notification code from this range implies failed EAP exchange, so the peer can use the notification as a failure indication. After receiving the EAP-Response/AKA-Notification for these notification codes, the server MUST send the EAP-Failure packet.

The receipt of a notification code with the F bit set to one (values 32768...65536) does not imply failure. Notification code 32768 has been reserved as a general notification code to indicate successful authentication.

The second most significant bit of the notification code is called the Phase bit (P bit). It specifies at which phase of the EAP-AKA exchange the notification can be used. If the P bit is set to zero, the notification can only be used after a successful EAP/AKA-Challenge round in full authentication or a successful EAP/AKA-Reauthentication round in reautentication. A re-authentication round is considered successful only if the peer has successfully verified AT_MAC and AT_COUNTER attributes, and does not include the AT_COUNTER_TOO_SMALL attribute in EAP-Response/AKA-Reauthentication.

If the P bit is set to one, the notification can only by used before the EAP/AKA-Challenge round in full authentication or before the EAP/AKA-Reauthentication round in reauthentication.

Section 6.10 and Section 6.11 specify what other attributes must be included in the notification packets.

Some of the notification codes are authorization related and hence not usually considered as part of the responsibility of an EAP method. However, they are included as part of EAP-AKA because there are currently no other ways to convey this information to the user in a localizable way, and the information is potentially useful for the user. An EAP-AKA server implementation may decide never to send these EAP-AKA notifications.

4.3.2 Result Indications

As discussed in Section 4.4, the server and the peer use explicit error messages in all error cases. If the server detects an error after successful authentication, the server uses an EAP-AKA notification to indicate failure to the peer. In this case, the result indication is integrity and replay protected.

By sending an EAP-Response/AKA-Challenge packet or an EAP-Response/AKA-Reauthentication packet (without AT_COUNTER_TOO_SMALL), the peer indicates that it has successfully authenticated the server and that the peer's local policy accepts the EAP exchange. In other words, these packets are implicit success indications from the peer to the server.

EAP-AKA also supports optional protected success indications from the server to the peer. If the EAP server wants to use protected success indications, it includes the AT_RESULT_IND attribute in the EAP-Request/AKA-Challenge or the EAP-Request/AKA-Reauthentication packet. This attribute indicates, that the EAP server would like to use result indications in both successful and unsuccessful cases. If the peer also wants this, the peer includes AT_RESULT_IND in EAP-Response/AKA-Challenge or EAP-Response/AKA-Re-authentication. The peer MUST NOT include AT_RESULT_IND if it did not receive AT_RESULT_IND from the server. If both the peer and the server used AT_RESULT_IND, then the EAP exchange is not complete yet, but an EAP-AKA notification round will follow. The following EAP-SIM notification may indicate either failure or success.

Success indications with the AT_NOTIFICATION code 32768 can only be used if both the server and the peer indicate they want to use them with AT_RESULT_IND. If the server did not include AT_RESULT_IND in

the EAP-Request/AKA-Challenge or EAP-Request/AKA-Reauthentication packet, or if the peer did not include AT_RESULT_IND in the corresponding response packet, then the server MUST NOT use protected success indications.

Because the AT_NOTIFICATION code 32768 is used to indicate success, the server MUST ignore the contents of the EAP-AKA response it receives to the EAP-Request/AKA-Notification with this code. Regardless of the contents of the EAP-AKA response, the server MUST send EAP-Success as the next packet.

4.4 Error Cases

This section specifies the operation of the peer and the server in error cases. The subsections below require the EAP-AKA peer and server to send an error packet (EAP-Response/AKA-Client-Error or EAP-Request/AKA-Notification) in error cases. However, implementations SHOULD NOT rely upon the correct error reporting behavior of the peer, authenticator, or the server. It is possible for error and other messages to be lost in transit or for a malicious participant to attempt to consume resources by not issuing error messages. Both the peer and the EAP server SHOULD have a mechanism to clean up state even if an error message or EAP-Success is not received after a timeout period.

4.4.1 Peer Operation

Two special error messages have been specified for error cases that are related to the processing of the UMTS AKA AUTN parameter, as described in Section 3: (1) if the peer does not accept AUTN, the peer responds with EAP-Response/AKA-Authentication-Reject (Section 6.5), and the server issues EAP-Failure, and (2) if the peer detects that the sequence number in AUTN is not correct, the peer responds with EAP-Response/AKA-Synchronization-Failure (Section 6.6), and the server proceeds with a new EAP-Request/AKA-Challenge.

In other error cases, when an EAP-AKA peer detects an error in a received EAP-AKA packet, the EAP-AKA peer responds with the EAP-Response/AKA-Client-Error packet. In response to the EAP-Response/AKA-Client-Error, the EAP server MUST issue the EAP-Failure packet and the authentication exchange terminates.

By default, the peer uses the client error code 0, "unable to process packet". This error code is used in the following cases:

- EAP exchange is not acceptable according to the peer's local policy.
- the peer is not able to parse the EAP request, i.e. the EAP request is malformed
- the peer encountered a malformed attribute
- wrong attribute types or duplicate attributes have been included in the EAP request

- a mandatory attribute is missing
- unrecognized non-skippable attribute
- unrecognized or unexpected EAP-AKA Subtype in the EAP request
- invalid AT_MAC
- invalid AT_CHECKCODE
- invalid pad bytes in AT_PADDING
- the peer does not want to process AT_PERMANENT_ID_REQ

4.4.2 Server Operation

If an EAP-AKA server detects an error in a received EAP-AKA response, the server MUST issue the EAP-Request/AKA-Notification packet with an AT_NOTIFICATION code that implies failure. By default, the server uses one of the general failure codes (0 or 16384). The choice between these two codes depends on the phase of the EAP-AKA exchange, see Section 4.3. The errors cases when the server issues an EAP-Request/AKA-Notification that implies failure include the following:

- the server is not able to parse the peer's EAP response
- the server encounters a malformed attribute, a non-recognized non-skippable attribute, or a duplicate attribute
- a mandatory attribute is missing or an invalid attribute was included
- unrecognized or unexpected EAP-AKA Subtype in the EAP Response
- invalid AT_MAC
- invalid AT_CHECKCODE
- invalid AT_COUNTER

4.4.3 EAP-Failure

The EAP-AKA server sends EAP-Failure in three cases:

1) In response to an EAP-Response/AKA-Client-Error packet the server has received from the peer, or

2) In response to an EAP-Response/AKA-Authentication-Reject packet the server has received from the peer, or

3) Following an EAP-AKA notification round, when the AT_NOTIFICATION code implies failure.

The EAP-AKA server MUST NOT send EAP-Failure in other cases than these three. However, it should be noted that even though the EAP-AKA server would not send an EAP-Failure, an authorization decision that happens outside EAP-AKA, such as in the AAA server or in an intermediate AAA proxy, may result in a failed exchange.

The peer MUST accept the EAP-Failure packet in case 1), case 2) and case 3) above. The peer SHOULD silently discard the EAP-Failure packet in other cases.

4.4.4 EAP-Success

On full authentication, the server can only send EAP-Success after the EAP/AKA-Challenge round. The peer MUST silently discard any EAP-Success packets if they are received before the peer has successfully authenticated the server and sent the EAP-Response/ AKA-Challenge packet.

If the peer did not indicate that it wants to use protected success indications with AT_RESULT_IND (as discussed in Section 4.3.2) on full authentication, then the peer MUST accept EAP-Success after a successful EAP/AKA-Challenge round.

If the peer indicated that it wants to use protected success indications with AT_RESULT_IND (as discussed in Section 4.3.2), then the peer MUST NOT accept EAP-Success after a successful EAP/ AKA-Challenge round. In this case, the peer MUST only accept EAP-Success after receiving an EAP-AKA Notification with the

AT_NOTIFICATION code 32768.

On fast re-authentication, EAP-Success can only be sent after the EAP/AKA-Reauthentication round. The peer MUST silently discard any EAP-Success packets if they are received before the peer has successfully authenticated the server and sent the EAP-Response/ AKA-Reauthentication packet.

If the peer did not indicate that it wants to use protected success indications with AT_RESULT_IND (as discussed in Section 4.3.2) on fast re-authentication, then the peer MUST accept EAP-Success after a successful EAP/AKA-Reauthentication round.

If the peer indicated that it wants to use protected success indications with AT_RESULT_IND (as discussed in Section 4.3.2), then the peer MUST NOT accept EAP-Success after a successful EAP/ AKA-Reauthentication round. In this case, the peer MUST only accept EAP-Success after receiving an EAP-AKA Notification with the AT_NOTIFICATION code 32768.

If the peer receives an EAP-AKA notification (Section 4.3) that indicates failure, then the peer MUST no longer accept the EAP-Success packet even if the server authentication was successfully completed.

4.5 Key Generation

This section specifies how keying material is generated.

On EAP-AKA full authentication, a Master Key (MK) is derived from the underlying UMTS AKA values (CK and IK keys), and the identity as follows.

MK = SHA1(Identity|IK|CK)

In the formula above, the "|" character denotes concatenation. Identity denotes the peer identity string without any terminating null characters. It is the identity from the AT_IDENTITY attribute from the last EAP-Response/AKA-Identity packet, or, if AT_IDENTITY was not used, the identity from the EAP-Response/Identity packet. The

identity string is included as-is, without any changes and including the possible identity decoration. The hash function SHA-1 is specified in [SHA-1].

The Master Key is fed into a Pseudo-Random number Function (PRF), which generates separate Transient EAP Keys (TEKs) for protecting EAP-AKA packets, as well as a Master Session Key (MSK) for link layer security and an Extended Master Session Key (EMSK) for other purposes. On fast re-authentication, the same TEKs MUST be used for protecting EAP packets, but a new MSK and a new EMSK MUST be derived from the original MK and new values exchanged in the fast re-authentication.

EAP-AKA requires two TEKs for its own purposes, the authentication key K_aut to be used with the AT_MAC attribute, and the encryption key K_encr, to be used with the AT_ENCR_DATA attribute. The same K_aut and K_encr keys are used in full authentication and subsequent fast re-authentications.

Key derivation is based on the random number generation specified in NIST Federal Information Processing Standards (FIPS) Publication 186-2 [PRF]. The pseudo-random number generator is specified in the change notice 1 (2001 October 5) of [PRF] (Algorithm 1). As specified in the change notice (page 74), when Algorithm 1 is used as a general-purpose pseudo-random number generator, the "mod q" term in step 3.3 is omitted. The function G used in the algorithm is constructed via Secure Hash Standard as specified in Appendix 3.3 of the standard. It should be noted that the function G is very similar to SHA-1, but the message padding is different. Please refer to [PRF] for full details. For convenience, the random number algorithm with the correct modification is cited in Annex A.

160-bit XKEY and XVAL values are used, so b = 160. On each full

authentication, the Master Key is used as the initial secret seed-key XKEY. The optional user input values (XSEED_j) in step 3.1 are set to zero.

On full authentication, the resulting 320-bit random numbers x_0, x_1, ..., x_m-1 are concatenated and partitioned into suitable-sized chunks and used as keys in the following order: K_encr (128 bits), K_aut (128 bits), Master Session Key (64 bytes), Extended Master

Session Key (64 bytes).

On fast re-authentication, the same pseudo-random number generator can be used to generate a new Master Session Key and a new Extended Master Session Key. The seed value XKEY' is calculated as follows:

XKEY' = SHA1(Identity|counter|NONCE_S| MK)

In the formula above, the Identity denotes the fast re-authentication identity, without any terminating null characters, from the AT_IDENTITY attribute of the EAP-Response/AKA-Identity packet, or, if EAP-Response/AKA-Identity was not used on fast re-authentication, the identity string from the EAP-Response/Identity packet. The counter denotes the counter value from AT_COUNTER attribute used in the EAP-Response/AKA-Reauthentication packet. The counter is used in network byte order. NONCE_S denotes the 16-byte random NONCE_S value from the AT_NONCE_S attribute used in the EAP-Request/ AKA-Reauthentication packet. The MK is the Master Key derived on the preceding full authentication.

On fast re-authentication, the pseudo-random number generator is run with the new seed value XKEY', and the resulting 320-bit random numbers x_0, x_1, ..., x_m-1 are concatenated and partitioned into 64-byte chunks and used as the new 64-byte Master Session Key and the new 64-byte Extended Master Session Key. Note that because K_encr and K_aut are not derived on fast re-authentication, the Master Session Key and the Extended Master Session key are obtained from the beginning of the key stream x_0, x_1, ...

The first 32 bytes of the MSK can be used as the Pairwise Master Key (PMK) for IEEE 802.11i.

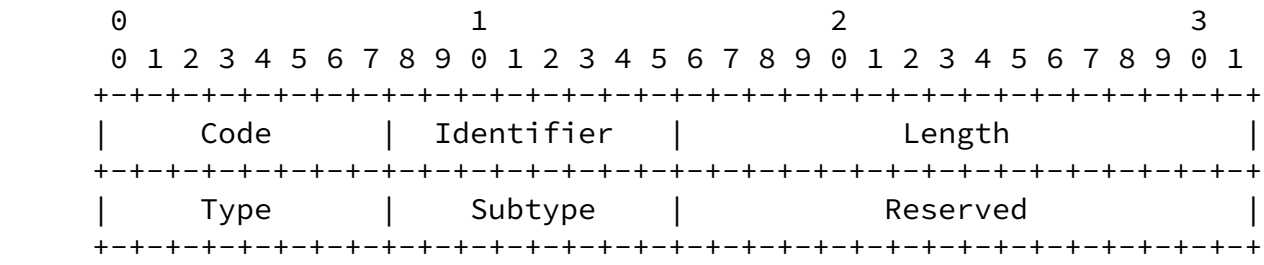
When the RADIUS attributes specified in [RFC2548] are used to transport keying material, then the first 32 bytes of the MSK correspond to MS-MPPE-RECV-KEY and the second 32 bytes to MS-MPPE-SEND-KEY. In this case, only 64 bytes of keying material (the MSK) are used.

5. Message Format and Protocol Extensibility

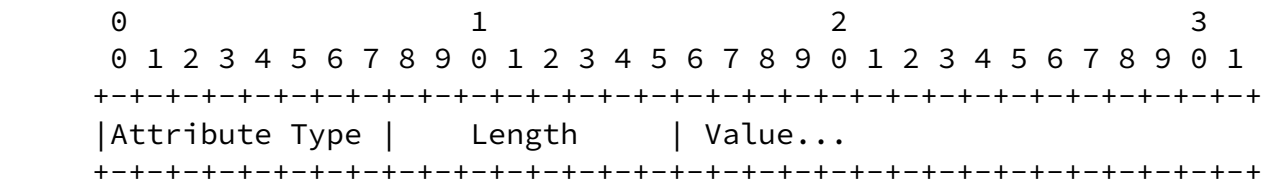
5.1 Message Format

As specified in [EAP], EAP packets begin with the Code, Identifiers, Length, and Type fields, which are followed by EAP method specific Type-Data. The Code field in the EAP header is set to 1 for EAP requests, and to 2 for EAP Responses. The usage of the Length and Identifier fields in the EAP header is also specified in [EAP]. In EAP-AKA, the Type field is set to 23.

In EAP-AKA, the Type-Data begins with an EAP-AKA header that consists of a 1-octet Subtype field, and a 2-octet reserved field. The Subtype values used in EAP-AKA are defined in Section 8. The formats of the EAP header and the EAP-AKA header are shown below.



The rest of the Type-Data, immediately following the EAP-AKA header, consists of attributes that are encoded in Type, Length, Value format. The figure below shows the generic format of an attribute.



Attribute Type

Indicates the particular type of attribute. The attribute type values are listed in Section 8.

Length

Indicates the length of this attribute in multiples of 4 bytes. The maximum length of an attribute is 1024 bytes. The length includes the Attribute Type and Length bytes.

Value

The particular data associated with this attribute. This field is always included and it is two or more bytes in length. The type and length fields determine the format and length of the value field.

Attributes numbered within the range 0 through 127 are called non-skippable attributes. When an EAP-AKA peer encounters a non-skippable attribute type that the peer does not recognize, the peer MUST send the EAP-Response/AKA-Client-Error packet, and the authentication exchange terminates. If an EAP-AKA server encounters a non-skippable attribute that the server does not recognize, then the server sends EAP-Request/AKA-Notification packet with an AT_NOTIFICATION code that implies general failure (0 or 16384 depending on the phase of the exchange), and the authentication exchange terminates.

When an attribute numbered in the range 128 through 255 is encountered but not recognized that particular attribute is ignored, but the rest of the attributes and message data MUST still be processed. The Length field of the attribute is used to skip the attribute value when searching for the next attribute. These attributes are called skippable attributes.

Unless otherwise specified, the order of the attributes in an EAP AKA message is insignificant, and an EAP-AKA implementation should not assume a certain order to be used.

Attributes can be encapsulated within other attributes. In other words, the value field of an attribute type can be specified to contain other attributes.

5.2 Protocol Extensibility

EAP-AKA can be extended by specifying new attribute types. If skippable attributes are used, it is possible to extend the protocol without breaking old implementations. As specified in Section 7.13, if new attributes are specified for EAP-Request/AKA-Identity or EAP-Response/AKA-Identity, then the AT_CHECKCODE MUST be used to integrity protect the new attributes.

When specifying new attributes, it should be noted that EAP-AKA does not support message fragmentation. Hence, the sizes of the new extensions MUST be limited so that the maximum transfer unit (MTU) of the underlying lower layer is not exceeded. According to [EAP], lower layers must provide an EAP MTU of 1020 bytes or greater, so any extensions to EAP-AKA SHOULD NOT exceed the EAP MTU of 1020 bytes.

EAP-AKA packets do not include a version field. However, should there

be a reason to revise this protocol in the future, new non-skippable or skippable attributes could be specified in order to implement revised EAP-AKA versions in a backward-compatible manner. It is possible to introduce version negotiation in the EAP-Request/ AKA-Identity and EAP-Response/AKA-Identity messages by specifying new skippable attributes.

6. Messages

This section specifies the messages used in EAP-AKA. It specifies when a message may be transmitted or accepted, which attributes are allowed in a message, which attributes are required in a message, and other message specific details. Message format is specified in Section 5.1.

6.1 EAP-Request/AKA-Identity

The EAP/AKA-Identity roundtrip MAY used for obtaining the peer identity to the server. As discussed in Section 4.1, several AKA-Identity rounds may be required in order to obtain a valid peer identity.

The server MUST include one of the following identity requesting attributes: AT_PERMANENT_ID_REQ, AT_FULLAUTH_ID_REQ, AT_ANY_ID_REQ. These three attributes are mutually exclusive, so the server MUST NOT include more than one of the attributes.

If the server has previously issued an EAP-Request/AKA-Identity message with the AT_PERMANENT_ID_REQ attribute, and if the server has received a response from the peer, then the server MUST NOT issue a new EAP-Request/AKA-Identity packet.

If the server has previously issued an EAP-Request/AKA-Identity message with the AT_FULLAUTH_ID_REQ attribute, and if the server has received a response from the peer, then the server MUST NOT issue a new EAP-Request/AKA-Identity packet with the AT_ANY_ID_REQ or AT_FULLAUTH_ID_REQ attributes.

If the server has previously issued an EAP-Request/AKA-Identity

message with the AT_ANY_ID_REQ attribute, and if the server has received a response from the peer, then the server MUST NOT issue a new EAP-Request/AKA-Identity packet with the AT_ANY_ID_REQ.

This message MUST NOT include AT_MAC, AT_IV, or AT_ENCR_DATA.

6.2 EAP-Response/AKA-Identity

The peer sends EAP-Response/AKA-Identity in response to a valid EAP-

Request/AKA-Identity from the server.

The peer MUST include the AT_IDENTITY attribute. The usage of AT_IDENITY is defined in Section 4.1.

This message MUST NOT include AT_MAC, AT_IV, or AT_ENCR_DATA.

6.3 EAP-Request/AKA-Challenge

The server sends the EAP-Request/AKA-Challenge on full authentication after successfully obtaining the subscriber identity.

The AT_RAND attribute MUST be included.

AT_MAC MUST be included. In EAP-Request/AKA-Challenge, there is no message-specific data covered by the MAC, see Section 7.15.

The AT_RESULT_IND attribute MAY be included. The usage of this attribute is discussed in Section 4.3.2.

The AT_CHECKCODE attribute MAY be included, and in certain cases specified in Section 7.13, it MUST be included.

The EAP-Request/AKA-Challenge packet MAY include encrypted attributes for identity privacy and for communicating the next re-authentication identity. In this case, the AT_IV and AT_ENCR_DATA attributes are included (Section 7.12).

The plaintext of the AT_ENCR_DATA value field consist of nested attributes. The nested attributes MAY include AT_PADDING (as specified in Section 7.12). If the server supports identity privacy and wants to communicate a pseudonym to the peer for the next full

authentication, then the nested encrypted attributes include the AT_NEXT_PSEUDONYM attribute. If the server supports re-authentication and wants to communicate a fast re-authentication identity to the peer, then the nested encrypted attributes include the AT_NEXT_REAUTH_ID attribute. Later versions of this protocol MAY specify additional attributes to be included within the encrypted data.

When processing this message, the peer MUST process AT_RAND and AT_AUTN before processing other attributes. Only if these attributes are verified to be valid, the peer derives keys and verifies AT_MAC. The operation in case an error occurs is specified in Section 4.4.1.

6.4 EAP-Response/AKA-Challenge

The peer sends EAP-Response/AKA-Challenge in response to a valid

EAP-Request/AKA-Challenge.

Sending this packet indicates, that the peer has successfully authenticated the server and that the EAP exchange will be accepted by the peer's local policy. Hence, if these conditions are not met, then the peer MUST NOT send EAP-Response/AKA-Challenge, but the peer MUST send EAP-Response/AKA-Client-Error.

The AT_MAC attribute MUST be included. In EAP-Response/AKA-Challenge, there is no message-specific data covered by the MAC, see Section 7.15.

The AT_RES attribute MUST be included.

The AT_CHECKCODE attribute MAY be included, and in certain cases specified in Section 7.13, it MUST be included.

The AT_RESULT_IND attribute MAY be included, if it was included in EAP-Request/AKA-Challenge. The usage of this attribute is discussed in Section 4.3.2.

Later versions of this protocol MAY make use of the AT_ENCR_DATA and AT_IV attributes in this message to include encrypted (skippable) attributes. The EAP server MUST process EAP-Response/AKA-Challenge messages that include these attributes even if the server did not

implement these optional attributes.

6.5 EAP-Response/AKA-Authentication-Reject

The peer sends the EAP-Response/AKA-Authentication-Reject packet if it does not accept the AUTN parameter. This version of the protocol does not specify any attributes for this message. Future versions of the protocol MAY specify attributes for this message.

The AT_MAC, AT_ENCR_DATA, or AT_IV attributes MUST NOT be used in this message.

6.6 EAP-Response/AKA-Synchronization-Failure

The peer sends the EAP-Response/AKA-Synchronization-Failure, when the sequence number in the AUTN parameter is incorrect.

The peer MUST include the AT_AUTS attribute. Future versions of the protocol MAY specify other additional attributes for this message.

The AT_MAC, AT_ENCR_DATA, or AT_IV attributes MUST NOT be used in this message.

6.7 EAP-Request/AKA-Reauthentication

The server sends the EAP-Request/AKA-Reauthentication message if it wants to use fast re-authentication, and if it has received a valid fast re-authentication identity in EAP-Response/Identity or EAP-Response/AKA-Identity.

The AT_MAC attribute MUST be included. No message-specific data is included in the MAC calculation, see Section 7.15.

The AT_RESULT_IND attribute MAY be included. The usage of this attribute is discussed in Section 4.3.2.

The AT_CHECKCODE attribute MAY be included, and in certain cases specified in Section 7.13, it MUST be included.

The AT_IV and AT_ENCR_DATA attributes MUST be included. The plaintext consists of the following nested encrypted attributes, which MUST be

included: AT_COUNTER and AT_NONCE_S. In addition, the nested encrypted attributes MAY include the following attributes: AT_NEXT_REAUTH_ID and AT_PADDING.

6.8 EAP-Response/AKA-Reauthentication

The client sends the EAP-Response/AKA-Reauthentication packet in response to a valid EAP-Request/AKA-Reauthentication.

The AT_MAC attribute MUST be included. For EAP-Response/AKA-Reauthentication, the MAC code is calculated over the following data: EAP packet| NONCE_S. The EAP packet is represented as specified in Section 5.1. It is followed by the 16-byte NONCE_S value from the server's AT_NONCE_S attribute.

The AT_CHECKCODE attribute MAY be included, and in certain cases specified in Section 7.13, it MUST be included.

The AT_IV and AT_ENCR_DATA attributes MUST be included. The nested encrypted attributes MUST include the AT_COUNTER attribute. The AT_COUNTER_TOO_SMALL attribute MAY be included in the nested encrypted attributes, and it is included in cases specified in Section 4.2. The AT_PADDING attribute MAY be included.

The AT_RESULT_IND attribute MAY be included, if it was included in EAP-Request/AKA-Reauthentication. The usage of this attribute is discussed in Section 4.3.2.

Sending this packet without AT_COUNTER_TOO_SMALL indicates, that the peer has successfully authenticated the server and that the EAP

exchange will be accepted by the peer's local policy. Hence, if these conditions are not met, then the peer MUST NOT send EAP-Response/AKA-Reauthentication, but the peer MUST send EAP-Response/AKA-Client-Error.

6.9 EAP-Response/AKA-Client-Error

The peer sends EAP-Response/AKA-Client-Error in error cases, as specified in Section 4.4.1.

The AT_CLIENT_ERROR_CODE attribute MUST be included. The AT_MAC,

AT_IV, or AT_ENCR_DATA attributes MUST NOT be used with this packet.

6.10 EAP-Request/AKA-Notification

The usage of this message is specified in Section 4.3.

The AT_NOTIFICATION attribute MUST be included.

The AT_MAC attribute MUST be included if the P bit of the AT_NOTIFICATION code is set to zero, and MUST NOT be included if the P bit is set to one. The P bit is discussed in in Section 4.3.

No message-specific data is included in the MAC calculation. See Section 7.15.

If EAP-Request/AKA-Notification is used on a fast re-authentication exchange, and if the P bit in AT_NOTIFICATION is set to zero, then AT_COUNTER is used for replay protection. In this case, the AT_ENCR_DATA and AT_IV attributes MUST be included, and the encapsulated plaintext attributes MUST include the AT_COUNTER attribute. The counter value included in AT_COUNTER MUST be the same as in the EAP-Request/AKA-Reauthentication packet on the same fast re-authentication exchange.

6.11 EAP-Response/AKA-Notification

The usage of this message is specified in Section 4.3. This packet is an acknowledgement of EAP-Request/AKA-Notification.

The AT_MAC attribute MUST included in cases when the P bit of the notification code in AT_NOTIFICATION of EAP-Request/AKA-Notification is set to zero, and MUST NOT be included in cases when the P bit is set to one. The P bit is discussed in Section 4.3.

If EAP-Request/AKA-Notification is used on fast a re-authentication exchange, and if the P bit in AT_NOTIFICATION is set to zero, then AT_COUNTER is used for replay protection. In this case, the

AT_ENCR_DATA and AT_IV attributes MUST be included, and the encapsulated plaintext attributes MUST include the AT_COUNTER attribute. The counter value included in AT_COUNTER MUST be the same as in the EAP-Request/AKA-Reauthentication packet on the same fast

re-authentication exchange.

7. Attributes

This section specifies the format of message attributes. The attribute type numbers are specified in Section 8.

7.1 Table of Attributes

The following table provides a guide to which attributes may be found in which kinds of messages, and in what quantity. Messages are denoted with numbers in parentheses as follows: (1) EAP-Request/ AKA-Identity, (2) EAP-Response/AKA-Identity, (3) EAP-Request/ AKA-Challenge, (4) EAP-Response/AKA-Challenge, (5) EAP-Request/ AKA-Notification, (6) EAP-Response/AKA-Notification, (7) EAP-Response/AKA-Client-Error (8) EAP-Request/AKA-Reauthentication, (9) EAP-Response/AKA-Re-authentication, (10) EAP-Response/ AKA-Authentication-Reject, and (11) EAP-Response/ AKA-Synchronization-Failure. The column denoted with "E" indicates whether the attribute is a nested attribute that MUST be included within AT_ENCR_DATA.

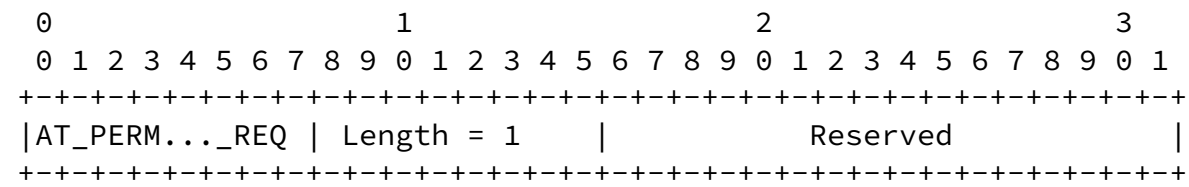
"0" indicates that the attribute MUST NOT be included in the message, "1" indicates that the attribute MUST be included in the message, "0-1" indicates that the attribute is sometimes included in the message, and "0*" indicates that the attribute is not included in the message in cases specified in this document, but MAY be included in the future versions of the protocol.

Attribute	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	E
AT_PERMANENT_ID_REQ	0-1	0	0	0	0	0	0	0	0	0	0	N
AT_ANY_ID_REQ	0-1	0	0	0	0	0	0	0	0	0	0	N
AT_FULLAUTH_ID_REQ	0-1	0	0	0	0	0	0	0	0	0	0	N
AT_IDENTITY	0	0-1	0	0	0	0	0	0	0	0	0	N
AT_RAND	0	0	1	0	0	0	0	0	0	0	0	N
AT_AUTN	0	0	1	0	0	0	0	0	0	0	0	N
AT_RES	0	0	0	1	0	0	0	0	0	0	0	N
AT_AUTS	0	0	0	0	0	0	0	0	0	0	1	N
AT_NEXT_PSEUDONYM	0	0	0-1	0	0	0	0	0	0	0	0	Y
AT_NEXT_REAUTH_ID	0	0	0-1	0	0	0	0	0-1	0	0	0	Y
AT_IV	0	0	0-1	0*	0-1	0-1	0	1	1	0	0	N
AT_ENCR_DATA	0	0	0-1	0*	0-1	0-1	0	1	1	0	0	N
AT_PADDING	0	0	0-1	0*	0-1	0-1	0	0-1	0-1	0	0	Y
AT_CHECKCODE	0	0	0-1	0-1	0	0	0	0-1	0-1	0	0	N
AT_RESULT_IND	0	0	0-1	0-1	0	0	0	0-1	0-1	0	0	N
AT_MAC	0	0	1	1	0-1	0-1	0	1	1	0	0	N
AT_COUNTER	0	0	0	0	0-1	0-1	0	1	1	0	0	Y
AT_COUNTER_TOO_SMALL	0	0	0	0	0	0	0	0	0-1	0	0	Y
AT_NONCE_S	0	0	0	0	0	0	0	1	0	0	0	Y
AT_NOTIFICATION	0	0	0	0	1	0	0	0	0	0	0	N
AT_CLIENT_ERROR_CODE	0	0	0	0	0	0	1	0	0	0	0	N

It should be noted that attributes AT_PERMANENT_ID_REQ, AT_ANY_ID_REQ and AT_FULLAUTH_ID_REQ are mutually exclusive, so that only one of them can be included at the same time. If one of the attributes AT_IV and AT_ENCR_DATA is included, then both of the attributes MUST be included.

7.2 AT_PERMANENT_ID_REQ

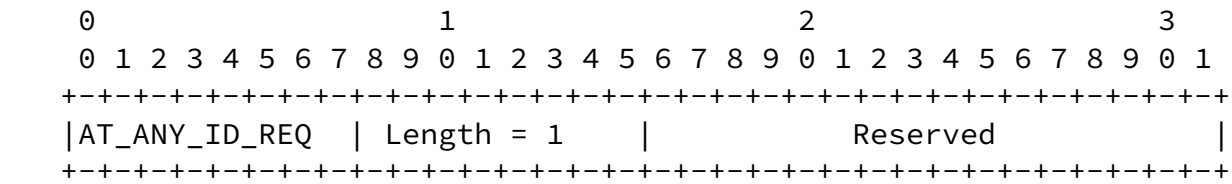
The format of the AT_PERMANENT_ID_REQ attribute is shown below.



The use of the AT_PERMANENT_ID_REQ is defined in Section 4.1. The value field only contains two reserved bytes, which are set to zero on sending and ignored on reception.

7.3 AT_ANY_ID_REQ

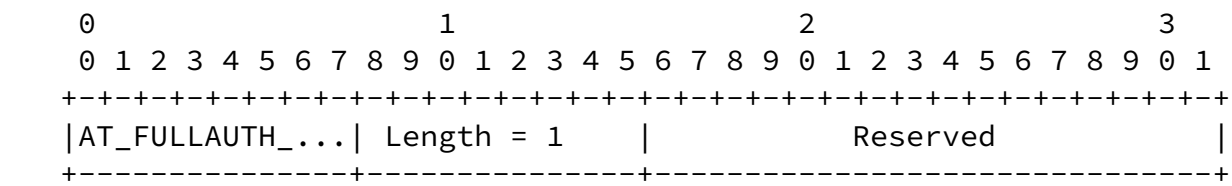
The format of the AT_ANY_ID_REQ attribute is shown below.



The use of the AT_ANY_ID_REQ is defined in Section 4.1. The value field only contains two reserved bytes, which are set to zero on sending and ignored on reception.

7.4 AT_FULLAUTH_ID_REQ

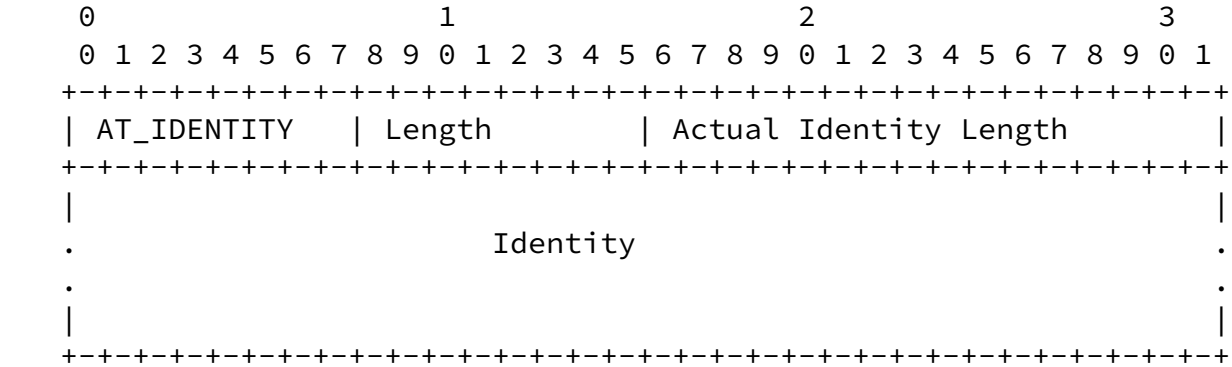
The format of the AT_FULLAUTH_ID_REQ attribute is shown below.



The use of the AT_FULLAUTH_ID_REQ is defined in Section 4.1. The value field only contains two reserved bytes, which are set to zero on sending and ignored on reception.

7.5 AT_IDENTITY

The format of the AT_IDENTITY attribute is shown below.



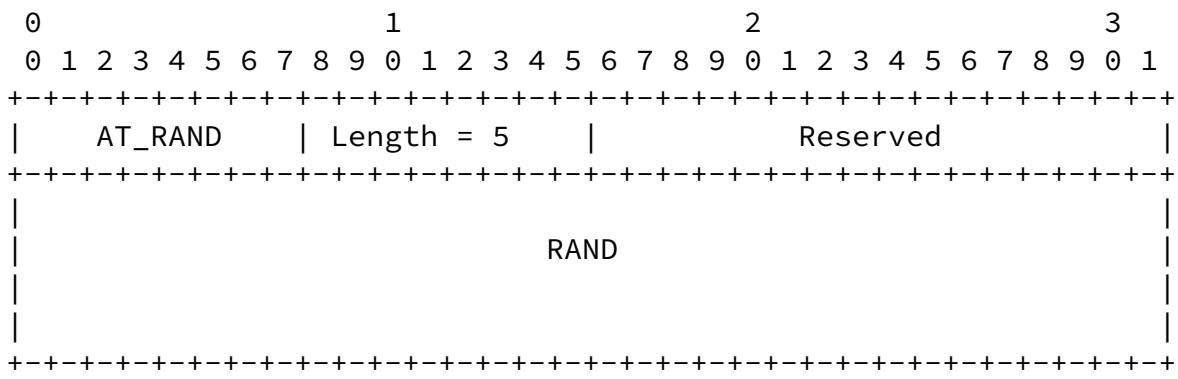
The use of the AT_IDENTITY is defined in Section 4.1. The value field

of this attribute begins with 2-byte actual identity length, which specifies the length of the identity in bytes. This field is followed by the subscriber identity of the indicated actual length. The identity is the permanent identity, a pseudonym identity or a fast re-authentication identity. The identity format is specified in Section 4.1.1. The same identity format is used in the AT_IDENTITY attribute and the EAP-Response/Identity packet, with the exception that the peer MUST NOT decorate the identity it includes in

AT_IDENTITY. The identity does not include any terminating null characters. Because the length of the attribute must be a multiple of 4 bytes, the sender pads the identity with zero bytes when necessary.

7.6 AT_RAND

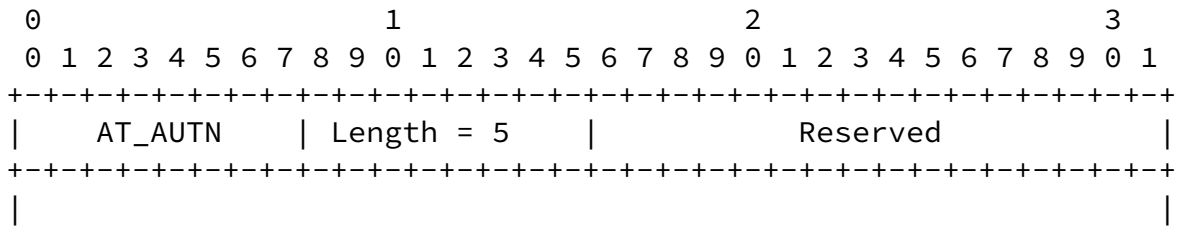
The format of the AT_RAND attribute is shown below.

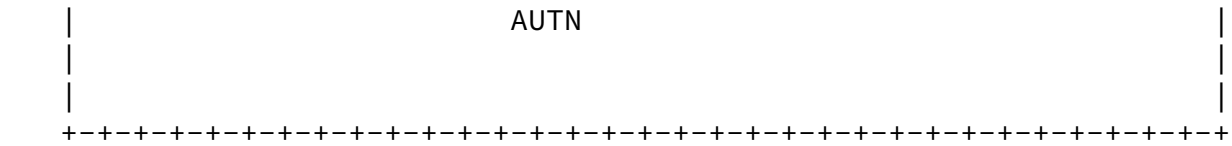


The value field of this attribute contains two reserved bytes followed by the AKA RAND parameter, 16 bytes (128 bits). The reserved bytes are set to zero when sending and ignored on reception.

7.7 AT_AUTN

The format of the AT_AUTN attribute is shown below.

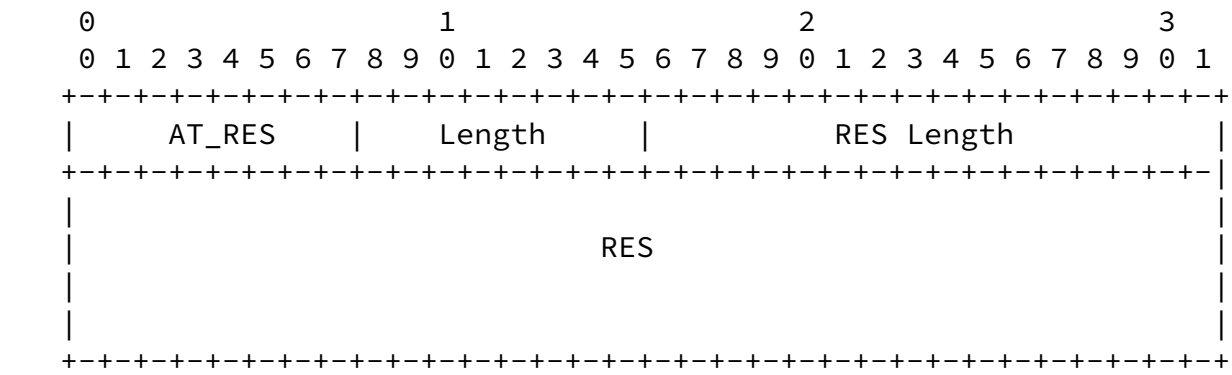




The value field of this attribute contains two reserved bytes followed by the AKA AUTN parameter, 16 bytes (128 bits). The reserved bytes are set to zero when sending and ignored on reception.

7.8 AT_RES

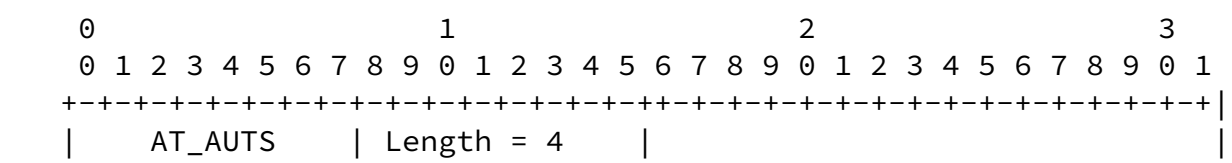
The format of the AT_RES attribute is shown below.

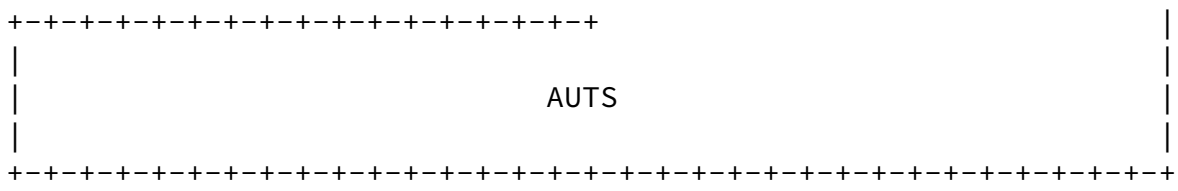


The value field of this attribute begins with the 2-byte RES Length, which is identifies the exact length of the RES in bits. The RES length is followed by the UMTS AKA RES parameter. According to [TS 33.105] the length of the AKA RES can vary between 32 and 128 bits. Because the length of the AT_RES attribute must be a multiple of 4 bytes, the sender pads the RES with zero bits where necessary.

7.9 AT_AUTS

The format of the AT_AUTS attribute is shown below.

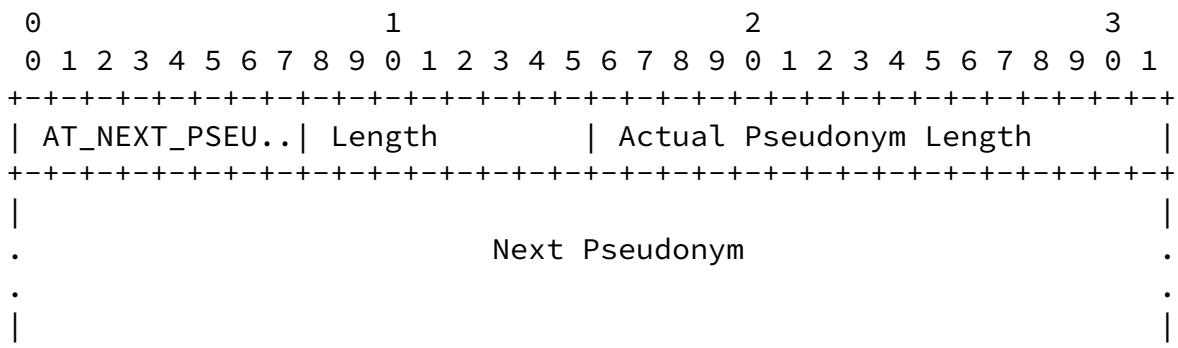




The value field of this attribute contains the AKA AUTS parameter, 112 bits (14 bytes).

7.10 AT_NEXT_PSEUDONYM

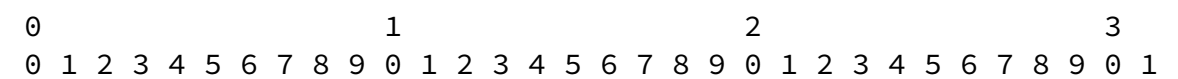
The format of the AT_NEXT_PSEUDONYM attribute is shown below.

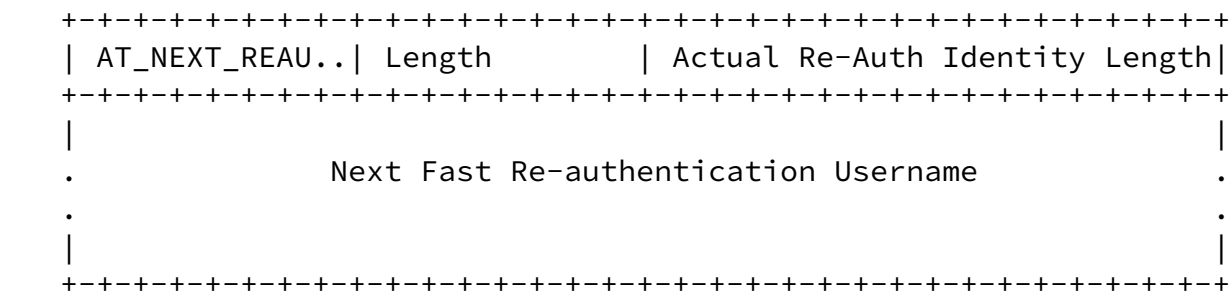


The value field of this attribute begins with 2-byte actual pseudonym length which specifies the length of the following pseudonym in bytes. This field is followed by a pseudonym username that the peer can use in the next authentication. The username MUST NOT include any realm portion. The username does not include any terminating null characters. Because the length of the attribute must be a multiple of 4 bytes, the sender pads the pseudonym with zero bytes when necessary. The username encoding MUST follow the UTF-8 transformation format [RFC2279]. This attribute MUST always be encrypted by encapsulating it within the AT_ENCR_DATA attribute.

7.11 AT_NEXT_REAUTH_ID

The format of the AT_NEXT_REAUTH_ID attribute is shown below.





The value field of this attribute begins with 2-byte actual re-authentication identity length which specifies the length of the following fast re-authentication identity in bytes. This field is followed by a fast re-authentication identity that the peer can use in the next fast re-authentication, as described in Section 4.2. In environments where a realm portion is required, the fast re-authentication identity includes both a username portion and a realm name portion. The fast re-authentication identity does not include any terminating null characters. Because the length of the attribute must be a multiple of 4 bytes, the sender pads the fast re-authentication identity with zero bytes when necessary. The identity encoding MUST follow the UTF-8 transformation format [RFC2279]. This attribute MUST always be encrypted by encapsulating it within the AT_ENCR_DATA attribute.

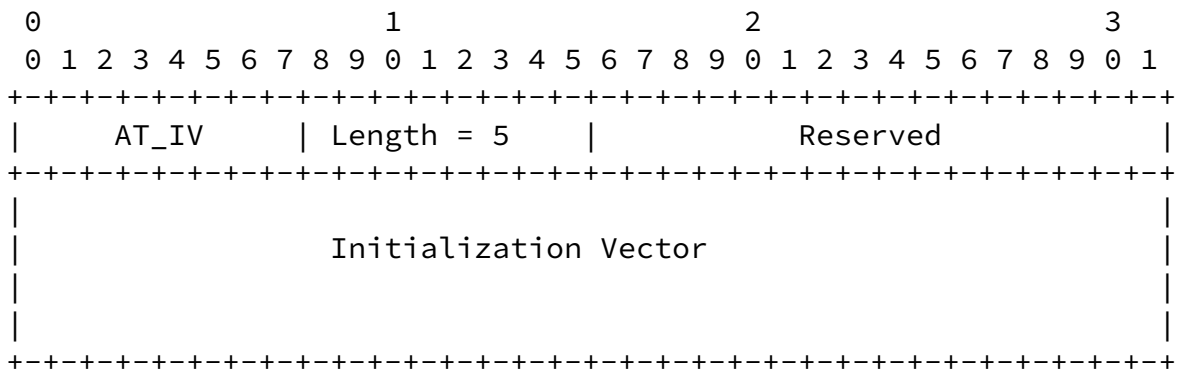
7.12 AT_IV, AT_ENCR_DATA and AT_PADDING

AT_IV and AT_ENCR_DATA attributes can be used to transmit encrypted information between the EAP-SIM peer and server.

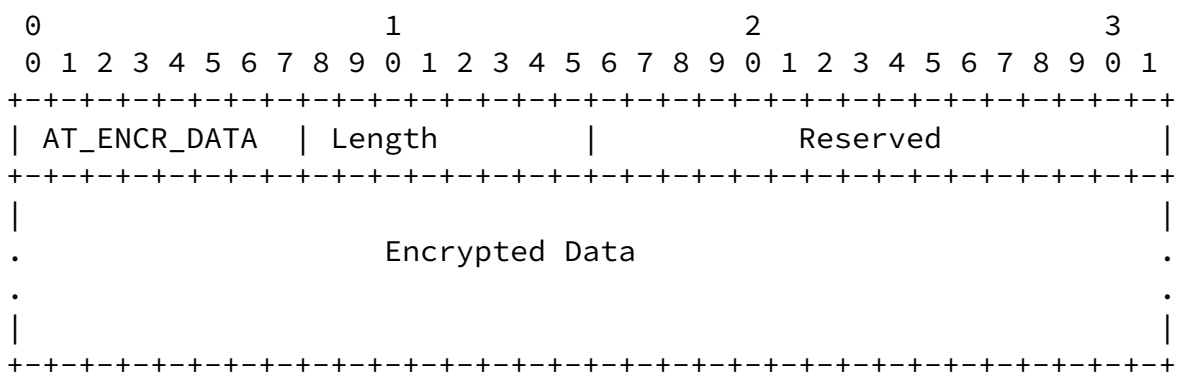
The value field of AT_IV contains two reserved bytes followed by a 16-byte initialization vector required by the AT_ENCR_DATA attribute. The reserved bytes are set to zero when sending and ignored on reception. The AT_IV attribute MUST be included if and only if the AT_ENCR_DATA is included. Section 4.4 specifies the operation if a packet that does not meet this condition is encountered.

The sender of the AT_IV attribute chooses the initialization vector by random. The sender MUST NOT reuse the initialization vector value from previous EAP-AKA packets. The sender SHOULD use a good source of randomness to generate the initialization vector. Please see [RFC1750] for more information about generating random numbers for

security applications. The format of AT_IV is shown below.



The value field of the AT_ENCR_DATA attribute consists of two reserved bytes followed by cipher text bytes encrypted using the Advanced Encryption Standard (AES) [AES] with a 128-bit key in the Cipher Block Chaining (CBC) mode of operation using the initialization vector from the AT_IV attribute. The reserved bytes are set to zero when sending and ignored on reception. Please see [CBC] for a description of the CBC mode. The format of the AT_ENCR_DATA attribute is shown below.

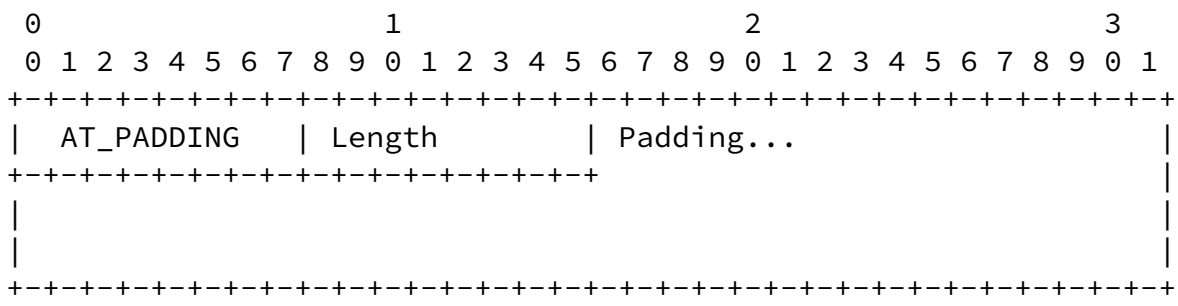


The derivation of the encryption key (K_encr) is specified in Section 4.5.

The plaintext consists of nested EAP-AKA attributes.

The encryption algorithm 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 AT_ENCR_DATA. The AT_PADDING

attribute is not included if the total length of other nested attributes within the AT_ENCR_DATA attribute is a multiple of 16 bytes. As usual, the Length of the Padding attribute includes the Attribute Type and Attribute Length fields. The length of the Padding attribute is 4, 8 or 12 bytes. It is chosen so that the length of the value field of the AT_ENCR_DATA attribute becomes a multiple of 16 bytes. The actual pad bytes in the value field are set to zero (00 hexadecimal) on sending. The recipient of the message MUST verify that the pad bytes are set to zero. If this verification fails on the peer, then it MUST send the EAP-Response/AKA-Client- Error packet with the error code "unable to process packet" to terminate the authentication exchange. If this verification fails on the server, then the server sends the EAP-Response/AKA-Notification packet with an AT_NOTIFICATION code that implies failure to terminate the authentication exchange. The format of the AT_PADDING attribute is shown below.

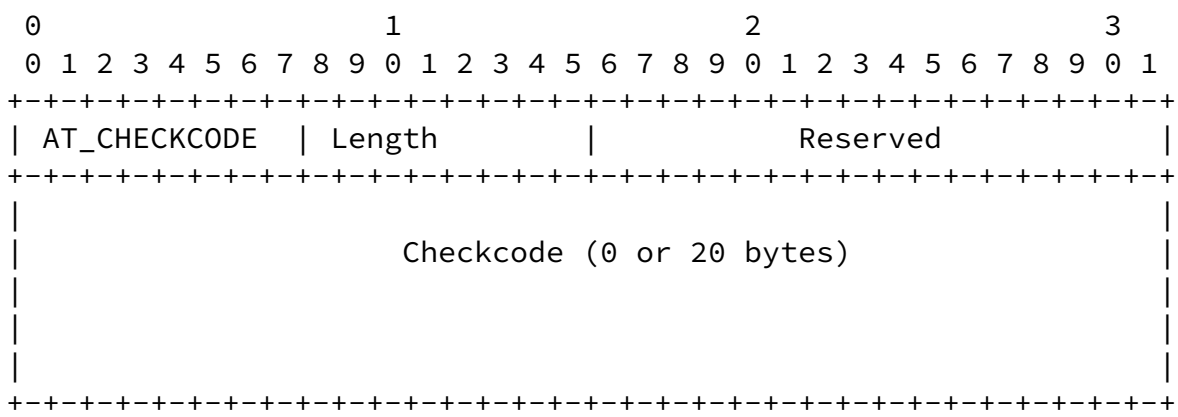


7.13 AT_CHECKCODE

The AT_MAC attribute is not used in the very first EAP-AKA messages during the AKA-Identity round, because keying material has not been derived yet. The peer and the server may exchange one or more pairs of EAP-AKA messages of the Subtype AKA-Identity before keys are derived and before the AT_MAC attribute can be applied. The EAP/ AKA-Identity messages may also be used upon fast re-authentication.

The AT_CHECKCODE attribute MAY be used to protect the EAP/ AKA-Identity messages. AT_CHECKCODE is included in EAP-Request/ AKA-Challenge and/or EAP-Response/AKA-Challenge upon full authentication. In fast re-authentication, AT_CHECKCODE MAY be included in EAP-Request/AKA-Reauthentication and/or EAP-Response/ AKA-Reauthentication. Because the AT_MAC attribute is used in these messages, AT_CHECKCODE will be integrity protected with AT_MAC. The

format of the AT_CHECKCODE attribute is shown below.



The value field of AT_CHECKCODE begins with two reserved bytes, which may be followed by a 20-byte checkcode. If the checkcode is not included in AT_CHECKCODE, then the attribute indicates that no EAP/ AKA-Identity messages were exchanged. This may occur in both full authentication and fast re-authentication. The reserved bytes are set to zero when sending and ignored on reception.

The checkcode is a hash value, calculated with SHA1 [SHA-1], over all EAP-Request/AKA-Identity and EAP-Response/ AKA-Identity packets exchanged in this authentication exchange. The packets are included in the order that they were transmitted, that is, starting with the first EAP-Request/ AKA-Identity message, followed by the corresponding EAP-Response/ AKA-Identity, followed by the second EAP-Request/ AKA-Identity (if used) etc.

EAP packets are included in the hash calculation "as-is", as they were transmitted or received. All reserved bytes, padding bytes etc. that are specified for various attributes are included as such, and the receiver must not reset them to zero. No delimiter bytes, padding or any other framing are included between the EAP packets when calculating the checkcode.

Messages are included in request/response pairs; in other words only full "round trips" are included. Packets that are silently discarded are not included. The EAP server must only include an EAP-Request/ AKA-Identity in the calculation once it has received a corresponding response, with the same Identifier value. Retransmissions or requests to which the server does not receive response are not included.

The peer must include the EAP-Request/AKA-Identity and the corresponding response in the calculation only if the peer receives a subsequent EAP-Request/AKA-Challenge, or a follow-up EAP-Request/ AKA-Identity with different attributes (attribute types) than in the first EAP-Request/AKA-Identity. After sending EAP-Response/

AKA-Identity, if the peer receives another EAP-Request/AKA-Identity with the same attributes as in the previous request, then the peer's response to the first request must have been lost. In this case the peer must not include the first request and its response in the calculation of the checkcode.

The AT_CHECKCODE attribute is optional to implement. It is specified in order to allow protecting the EAP/AKA-Identity messages and any future extensions to them. The implementation of AT_CHECKCODE is RECOMMENDED.

If the receiver of AT_CHECKCODE implements this attribute, then the receiver MUST check that the checkcode is correct. If the checkcode is invalid, the receiver must operate as specified in Section 4.4.

If the EAP/AKA-Identity messages are extended with new attributes then AT_CHECKCODE MUST be implemented and used. More specifically, if the server includes any other attributes than AT_PERMANENT_ID_REQ, AT_FULLAUTH_ID_REQ or AT_ANY_ID_REQ in the EAP-Request/AKA-Identity packet, then the server MUST include AT_CHECKCODE in EAP-Request/ AKA-Challenge or EAP-Request/AKA-Reauthentication. If the peer includes any other attributes than AT_IDENTITY in the EAP-Response/ AKA-Identity message, then the peer MUST include AT_CHECKCODE in EAP-Response/AKA-Challenge or EAP-Response/AKA-Reauthentication.

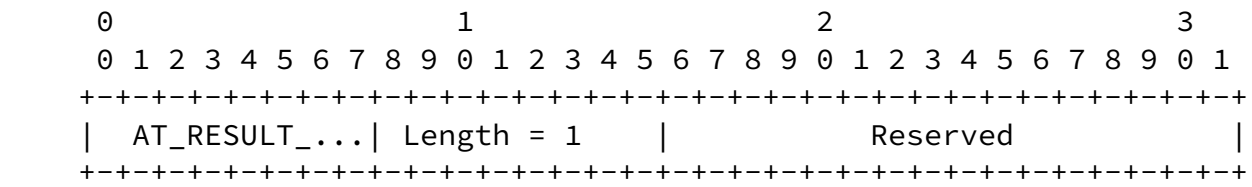
If the server implements the processing of any other attribute than AT_IDENTITY for the EAP-Response/AKA-Identity message, then the server MUST implement AT_CHECKCODE. In this case, if the server receives any other attribute than AT_IDENTITY in the EAP- Response/ AKA-Identity message, then the server MUST check that AT_CHECKCODE is present in EAP-Response/AKA-Challenge or EAP- Response/ AKA-Reauthentication. The operation when a mandatory attribute is missing is specified in Section 4.4.

Similarly, if the peer implements the processing of any other attribute than AT_PERMANENT_ID_REQ, AT_FULLAUTH_ID_REQ or AT_ANY_ID_REQ for the EAP-Request/AKA-Identity packet, then the peer MUST implement AT_CHECKCODE. In this case, if the peer receives any other attribute than AT_PERMANENT_ID_REQ, AT_FULLAUTH_ID_REQ or AT_ANY_ID_REQ in the EAP-Request/AKA-Identity packet, then the peer MUST check that AT_CHECKCODE is present in EAP-Request/AKA-Challenge or EAP-Request/AKA-Reauthentication. The operation when a mandatory

attribute is missing is specified in Section 4.4.

7.14 AT_RESULT_IND

The format of the AT_RESULT_IND attribute is shown below.



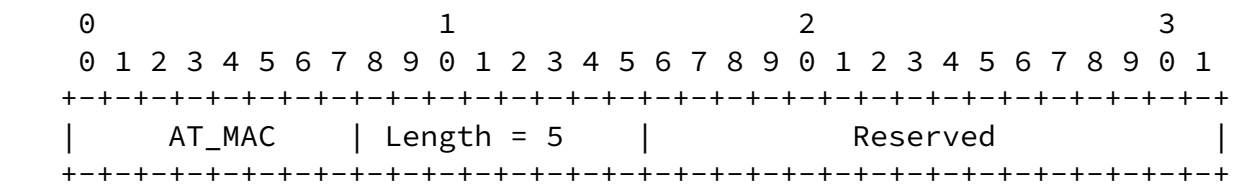
The value field of this attribute consists of two reserved bytes, which are set to zero upon sending and ignored upon reception. This attribute is always sent unencrypted, so it MUST NOT be encapsulated within the AT_ENCR_DATA attribute.

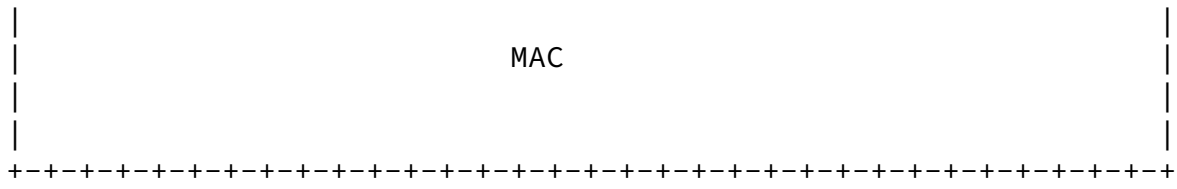
7.15 AT_MAC

The AT_MAC attribute is used for EAP-AKA message authentication. Section 6 specifies which messages AT_MAC MUST be included.

The value field of the AT_MAC attribute contains two reserved bytes followed by a keyed message authentication code (MAC). The MAC is calculated over the whole EAP packet, concatenated with optional message-specific data, with the exception that the value field of the MAC attribute is set to zero when calculating the MAC. The EAP packet includes the EAP header that begins with the Code field, the EAP-AKA header that begins with the Subtype field, and all the attributes, as specified in Section 5.1. The reserved bytes in AT_MAC are set to zero when sending and ignored on reception. The contents of the message-specific data that may be included in the MAC calculation are specified separately for each EAP-AKA message in Section 6.

The format of the AT_MAC attribute is shown below.





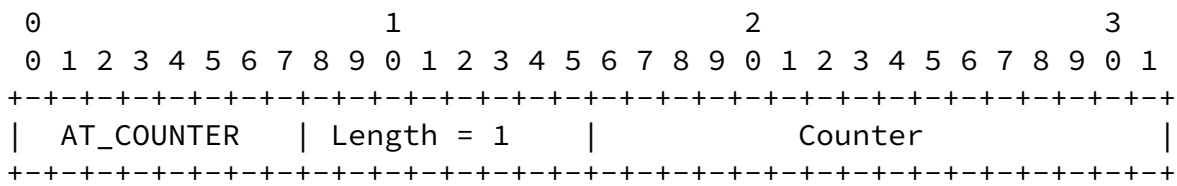
The MAC algorithm is HMAC-SHA1-128 [RFC2104] keyed hash value. (The HMAC-SHA1-128 value is obtained from the 20-byte HMAC-SHA1 value by truncating the output to 16 bytes. Hence, the length of the MAC is 16 bytes.) The derivation of the authentication key (K_aut) used in the calculation of the MAC is specified in Section 4.5.

When the AT_MAC attribute is included in an EAP-AKA message, the

recipient MUST process the AT_MAC attribute before looking at any other attributes, except when processing EAP-Request/AKA-Challenge. The processing of EAP-Request/AKA-Challenge is specified in Section 6.3. If the message authentication code is invalid, then the recipient MUST ignore all other attributes in the message and operate as specified in Section 4.4.

7.16 AT_COUNTER

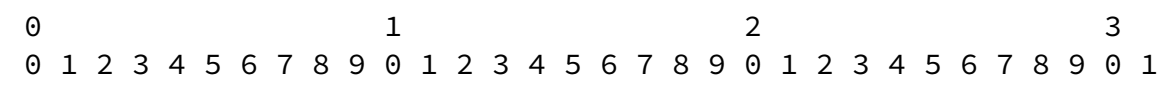
The format of the AT_COUNTER attribute is shown below.

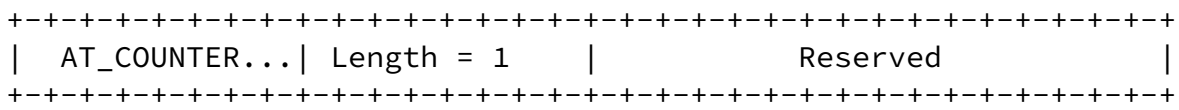


The value field of the AT_COUNTER attribute consists of a 16-bit unsigned integer counter value, represented in network byte order. This attribute MUST always be encrypted by encapsulating it within the AT_ENCR_DATA attribute.

7.17 AT_COUNTER_TOO_SMALL

The format of the AT_COUNTER_TOO_SMALL attribute is shown below.

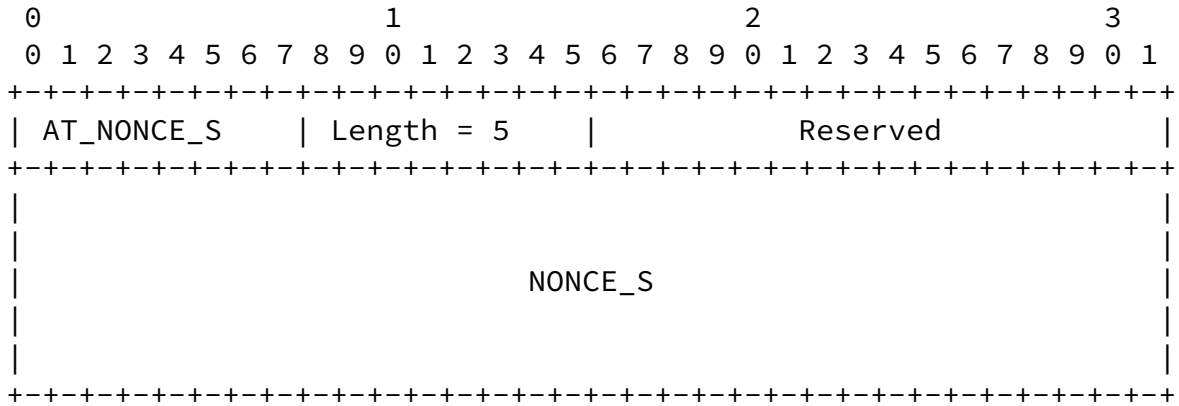




The value field of this attribute consists of two reserved bytes, which are set to zero upon sending and ignored upon reception. This attribute MUST always be encrypted by encapsulating it within the AT_ENCR_DATA attribute.

7.18 AT_NONCE_S

The format of the AT_NONCE_S attribute is shown below.



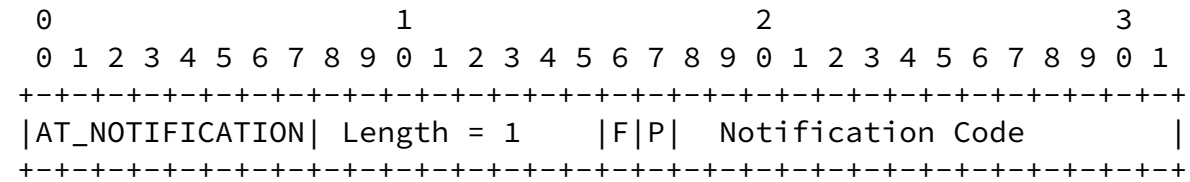
The value field of the AT_NONCE_S attribute contains two reserved bytes followed by a random number generated by the server (16 bytes) freshly for this EAP-AKA fast re-authentication. The random number is used as challenge for the peer and also a seed value for the new keying material. The reserved bytes are set to zero upon sending and ignored upon reception. This attribute MUST always be encrypted by encapsulating it within the AT_ENCR_DATA attribute.

The server MUST NOT reuse the NONCE_S value from a previous EAP-AKA

fast re-authentication exchange. The server SHOULD use a good source of randomness to generate NONCE_S. Please see [RFC1750] for more information about generating random numbers for security applications.

7.19 AT_NOTIFICATION

The format of the AT_NOTIFICATION attribute is shown below.



The value field of this attribute contains a two-byte notification code. The first and second bit (F and P) of the notification code are interpreted as described in Section 4.3.

The notification code values listed below have been reserved. The descriptions below illustrate the semantics of the notifications. The peer implementation MAY use different wordings when presenting the notifications to the user. The "requested service" depends on the environment where EAP-AKA is applied.

0 - General failure. (implies failure, used after successful authentication)

16384 - General failure. (implies failure, used before authentication)

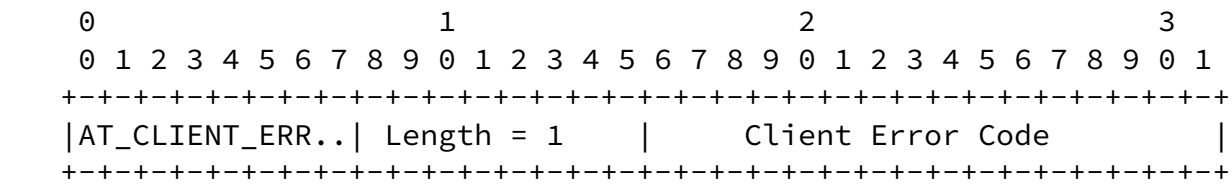
32768 - User has been successfully authenticated. (does not imply failure, used after successful authentication). The usage of this code is discussed in Section 4.3.2.

1026 - User has been temporarily denied access to the requested service. (Implies failure, used after successful authentication)

1031 - User has not subscribed to the requested service (implies failure, used after successful authentication)

7.20 AT_CLIENT_ERROR_CODE

The format of the AT_CLIENT_ERROR_CODE attribute is shown below.



The value field of this attribute contains a two-byte client error code. The following error code values have been reserved.

0 "unable to process packet": a general error code

8. IANA and Protocol Numbering Considerations

IANA has assigned the EAP type number 23 for EAP-AKA authentication.

EAP-AKA messages include a Subtype field. The Subtype is a new numbering space for which IANA administration is required. The following Subtypes are specified in this document:

AKA-Challenge.....1
AKA-Authentication-Reject.....2
AKA-Synchronization-Failure.....4
AKA-Identity.....5
AKA-Notification.....12
AKA-Reauthentication.....13
AKA-Client-Error.....14

The messages are composed of attributes, which have attribute type numbers. The EAP-AKA attribute type number is a new numbering space for which IANA administration is required. The following attribute types are specified in this document:

AT_RAND.....1
AT_AUTN.....2
AT_RES.....3
AT_AUTS.....4
AT_PADDING.....6
AT_PERMANENT_ID_REQ.....10
AT_MAC.....11

AT_NOTIFICATION.....12
AT_ANY_ID_REQ.....13
AT_IDENTITY.....14
AT_FULLAUTH_ID_REQ.....17
AT_COUNTER.....19
AT_COUNTER_TOO_SMALL.....20
AT_NONCE_S.....21
AT_CLIENT_ERROR_CODE.....22
AT_IV.....129
AT_ENCR_DATA.....130
AT_NEXT_PSEUDONYM.....132
AT_NEXT_REAUTH_ID.....133
AT_CHECKCODE.....134
AT_RESULT_IND.....135

The AT_NOTIFICATION attribute contains a notification code value. The notification code is a new numbering space for which IANA administration is required. Values 0, 1024, 1026, 1031, 16384 and 32768 have been specified in Section 7.19 of this document.

The AT_CLIENT_ERROR_CODE attribute contains a client error code. The client error code is a new numbering space for which IANA administration is required. Value 0 has been specified in Section 7.20 of this document.

All requests for value assignment from the various number spaces described in this document require proper documentation, according to the "Specification Required" policy described in [RFC2434]. Requests must be specified in sufficient detail so that interoperability between independent implementations is possible. Possible forms of documentation include, but are not limited to, RFCs, the products of another standards body (e.g. 3GPP), or permanently and readily available vendor design notes.

EAP-AKA and EAP-SIM [EAP-SIM] are "sister" protocols with similar message structure and protocol numbering spaces. Many attributes and message Subtypes have the same protocol numbers in these two protocols. Hence, it is recommended that the same protocol number value SHOULD NOT be allocated for two different purposes in EAP-AKA and EAP-SIM.

9. Security Considerations

The EAP base protocol specification [EAP] highlights several attacks that are possible against the EAP protocol. This section discusses the claimed security properties of EAP-AKA as well as vulnerabilities and security recommendations.

9.1 Identity Protection

EAP-AKA includes optional Identity privacy support that protects the privacy of the subscriber identity against passive eavesdropping. This document only specifies a mechanism to deliver pseudonyms from the server to the peer as part of an EAP-SIM exchange. Hence, a peer that has not yet performed any EAP-SIM exchanges does not typically have a pseudonym available. If the peer does not have a pseudonym available, then the privacy mechanism cannot be used, but the permanent identity will have to be sent in the clear. The terminal SHOULD store the pseudonym in a non-volatile memory so that it can be maintained across reboots. An active attacker that impersonates the network may use the AT_PERMANENT_ID_REQ attribute (Section 4.1.2) to learn the subscriber's IMSI. However, as discussed in Section 4.1.2, the terminal can refuse to send the cleartext IMSI if it believes that the network should be able to recognize the pseudonym.

If the peer and server cannot guarantee that the pseudonym will be maintained reliably and Identity privacy is required then additional protection from an external security mechanism such as Protected Extensible Authentication Protocol (PEAP) [PEAP] may be used. The benefits and the security considerations of using an external security mechanism with EAP-AKA are beyond the scope of this document.

9.2 Mutual Authentication

EAP-AKA provides mutual authentication via the UMTS AKA mechanisms.

9.3 Flooding the Authentication Centre

The EAP-AKA server typically obtains authentication vectors from the Authentication Centre (AuC). EAP-AKA introduces a new usage for the AuC. The protocols between the EAP-AKA server and the AuC are out of the scope of this document. However, it should be noted that a malicious EAP-AKA peer may generate a lot of protocol requests to mount a denial of service attack. The EAP-AKA server implementation SHOULD take this into account and SHOULD take steps to limit the traffic that it generates towards the AuC, preventing the attacker from flooding the AuC and from extending the denial of service attack from EAP-AKA to other users of the AuC.

9.4 Key Derivation

EAP-AKA supports key derivation with 128-bit effective key strength. The key hierarchy is specified in Section 4.5.

The Transient EAP Keys used to protect EAP-AKA packets (K_encr, K_aut) and the Master Session Keys are cryptographically separate. An attacker cannot derive any non-trivial information from K_encr or K_aut based on the Master Session Key or vice versa. An attacker also cannot calculate the pre-shared secret from the UMTS AKA IK, UMTS AKA CK, EAP-AKA K_encr, EAP-AKA K_aut or from the Master Session Key.

9.5 Brute-Force and Dictionary Attacks

The effective strength of EAP-AKA values is 128 bits, and there are no known computationally feasible brute-force attacks. Because UMTS AKA is not a password protocol (the pre-shared secret must not be a weak password), EAP-AKA is not vulnerable to dictionary attacks.

9.6 Protection, Replay Protection and Confidentiality

AT_MAC, AT_IV, AT_ENCR_DATA and AT_COUNTER attributes are used to provide integrity, replay and confidentiality protection for EAP-AKA Requests and Responses. Integrity protection with AT_MAC includes the EAP header. Integrity protection (AT_MAC) is based on a keyed message authentication code. Confidentiality (AT_ENCR_DATA and AT_IV) is based on a block cipher.

Because keys are not available in the beginning of the EAP methods, the AT_MAC attribute cannot be used for protecting EAP/AKA-Identity messages. However, the AT_CHECKCODE attribute can optionally be used to protect the integrity of the EAP/AKA-Identity roundtrip.

Confidentiality protection is applied only to a part of the protocol fields. The table of attributes in Section 7.1 summarizes which fields are confidentiality protected. It should be noted that the error and notification code attributes AT_CLIENT_ERROR_CODE and AT_NOTIFICATION are not confidential but they are transmitted in the clear. Identity protection is discussed in Section 9.1.

On full authentication, replay protection of the EAP exchange is provided by RAND and AUTN values from the underlying UMTS AKA scheme. Protection against replays of EAP-AKA messages is also based on the

fact that messages that can include AT_MAC can only be sent once with a certain EAP-AKA Subtype, and on the fact that a different K_aut key will be used for calculating AT_MAC in each full authentication exchange.

On fast re-authentication, a counter included in AT_COUNTER and a server random nonce is used to provide replay protection. The AT_COUNTER attribute is also included in EAP-AKA notifications, if they are used after successful authentication in order to provide replay protection between re-authentication exchanges.

The contents of the user identity string are implicitly integrity protected by including them in key derivation.

Because EAP-AKA is not a tunneling method, EAP-Request/Notification, EAP-Response/Notification, EAP-Success or EAP-Failure packets are not confidential, integrity protected or replay protected. On physically insecure networks, this may enable an attacker to mount denial of service attacks by spoofing these packets. As discussed in Section 4.4, the peer will only accept EAP-Success after successful authentication. Hence, the attacker cannot force the peer to believe successful authentication has occurred when mutual authentication failed or has not happened yet.

The security considerations of EAP-AKA result indications are covered in Section 9.8

An eavesdropper will see the EAP Notification, EAP_Success and EAP-Failure packets sent in the clear. With EAP-AKA, confidential information MUST NOT be transmitted in EAP Notification packets.

9.7 Negotiation Attacks

EAP-AKA does not protect the EAP-Response/Nak packet. Because EAP-AKA does not protect the EAP method negotiation, EAP method downgrading attacks may be possible, especially if the user uses the same identity with EAP-AKA and other EAP methods.

As described in Section 5, EAP-AKA allows the protocol to be extended by defining new attribute types. When defining such attributes, it should be noted that any extra attributes included in EAP-Request/

AKA-Identity or EAP-Response/AKA-Identity packets are not included in the MACs later on, and thus some other precautions must be taken to avoid modifications to them.

EAP-AKA does not support ciphersuite negotiation or EAP-AKA protocol version negotiation.

9.8 Protected Result Indications

EAP-AKA supports optional protected success indications, and acknowledged failure indications. If a failure occurs after successful authentication, then the EAP-AKA failure indication is

integrity and replay protected.

Even if an EAP-Failure packet is lost when using EAP-SIM over an unreliable medium, then the EAP-SIM failure indications will help ensure that the peer and EAP server will know the other parties authentication decision. If protected success indications are used, then the loss of Success packet will also be addressed by the acknowledged, integrity and replay protected EAP-SIM success indication. If the optional success indications are not used, then the peer may end up believing the server succeeded authentication when it actually failed. Since access will not be granted in this case protected result indications are not needed unless the client is not able to realize it does not have access for an extended period of time.

9.9 Man-in-the-middle Attacks

In order to avoid man-in-the-middle attacks and session hijacking, user data SHOULD be integrity protected on physically insecure networks. The EAP-AKA Master Session Key or keys derived from it MAY be used as the integrity protection keys, or, if an external security mechanism such as PEAP is used, then the link integrity protection keys MAY be derived by the external security mechanism.

There are man-in-the-middle attacks associated with the use of any EAP method within a tunneled protocol such as PEAP, or within a sequence of EAP methods followed by each other. This specification does not address these attacks. If EAP-AKA is used with a tunneling protocol or as part of a sequence of methods, there should be

cryptographic binding provided between the protocols and EAP-AKA to prevent man-in-the-middle attacks through rogue authenticators being able to setup one-way authenticated tunnels. EAP-AKA Master Session Key MAY be used to provide the cryptographic binding. However the mechanism how the binding is provided depends on the tunneling or sequencing protocol, and it is beyond the scope of this document.

9.10 Generating Random Numbers

An EAP-AKA implementation SHOULD use a good source of randomness to generate the random numbers required in the protocol. Please see [RFC1750] for more information on generating random numbers for security applications.

10. Security Claims

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

Auth. Mechanism: EAP-AKA is based on the UMTS AKA mechanism, which is

an authentication and key agreement mechanism based on a symmetric 128-bit pre-shared secret.

Ciphersuite negotiation: No

Mutual authentication: Yes

Integrity protection: Yes (Section 9.6)

Replay protection: Yes (Section 9.6)

Confidentiality: Yes, except method specific success and failure indications (Section 9.1, Section 9.6)

Key derivation: Yes

Key strength: EAP-AKA supports key derivation with 128-bit effective key strength.

Description of key hierarchy: Please see Section 4.5.

Dictinary attack protection: N/A (Section 9.5)

Fast reconnect: Yes

Cryptographic binding: N/A

Session independence: Yes (Section 9.4)

Fragmentation: No

Channel binding: No

Indication of vulnerabilities. Vulnerabilities are discussed in Section 9.

11. Acknowledgements and Contributions

The authors wish to thank Rolf Blom of Ericsson, Bernard Aboba of Microsoft, Arne Norefors of Ericsson, N.Asokan of Nokia, Valtteri Niemi of Nokia, Kaisa Nyberg of Nokia, Jukka-Pekka Honkanen of Nokia, Pasi Eronen of Nokia, Olivier Paridaens of Alcatel and Ilkka Uusitalo of Ericsson for interesting discussions in this problem space.

This protocol has been partly developed in parallel with EAP-SIM [EAP-SIM], and hence this specification incorporates many ideas from EAP-SIM, and many contributions from the reviewer's of EAP-SIM.

The attribute format is based on the extension format of Mobile IPv4 [RFC3344].

Normative References

[TS 33.102] 3rd Generation Partnership Project, "3GPP Technical Specification 3GPP TS 33.102 V5.1.0: "Technical Specification Group Services and System Aspects; 3G Security; Security Architecture (Release 5)"", December 2002.

[RFC2486] Aboba, B. and M. Beadles, "The Network Access Identifier", RFC 2486, January 1999.

[EAP] Blunk, L., Vollbrecht, J., Aboba, B., Carlson, J. and H. Levkowetz, "Extensible Authentication Protocol (EAP)", draft-ietf-eap-rfc2284bis-09 (work in progress), February 2004.

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

[TS 23.003] 3rd Generation Partnership Project, "3GPP Technical Specification 3GPP TS 23.003 V5.5.1: "3rd Generation Parnership Project; Technical Specification Group Core Network; Numbering, addressing and identification (Release 5)"", January 2003.

[RFC2104] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[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/nistpubs/800-38a/sp800-38a.pdf

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

[PRF] National Institute of Standards and Technology, "Federal Information Processing Standards (FIPS) Publication 186-2 (with change notice); Digital Signature Standard (DSS)",

January 2000.

Available on-line at: http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf

[TS 33.105] 3rd Generation Partnership Project, "3GPP Technical Specification 3GPP TS 33.105 4.1.0: "Technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements (Release 4)"", June 2001.

[RFC2279] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

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

Informative References

[RFC2548] Zorn, G., "Microsoft Vendor-specific RADIUS Attributes", RFC 2548, March 1999.

[PEAP] Palekar, A., Simon, D., Zorn, G., Salowey, J., Zhou, H. and S. Josefsson, "Protected EAP Protocol (PEAP)", draft-josefsson-pppext-eap-tls-eap-07 (work in progress), October 2003.

[RFC1750] Eastlake, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[RFC3344] Perkins, C., "IP Mobility Support for IPv4", RFC 3344, August 2002.

[EAP-SIM] Haverinen, H. and J. Salowey, "Extensible Authentication Protocol Method for GSM Subscriber Identity Modules (EAP-SIM)", draft-haverinen-pppext-eap-sim-13 (work in progress), April 2004.

[Draft 3GPP TS 23.003]

3rd Generation Partnership Project, "Draft 3GPP Technical Specification 3GPP TS 23.003 V 6.1.0: "3rd Generation Partnership Project; Technical Specification Group Core Network; Numbering, addressing and identification (Release 6)", December 2003.

work in progress

Authors' Addresses

Jari Arkko
Ericsson
FIN-02420 Jorvas
Finland

Phone: +358 40 5079256
EMail: jari.Arkko@ericsson.com

Henry Haverinen
Nokia Enterprise Solutions
P.O. Box 12
FIN-40101 Jyvaskyla
Finland

EMail: henry.haverinen@nokia.com

Appendix A. Pseudo-Random Number Generator

The "|" character denotes concatenation, and "^" denotes exponentiation.

Step 1: Choose a new, secret value for the seed-key, XKEY

Step 2: In hexadecimal notation let
    t = 67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0
    This is the initial value for H0|H1|H2|H3|H4
    in the FIPS SHS <xref target="SHA-1"/>

Step 3: For j = 0 to m - 1 do
      3.1 XSEED_j = 0 /* no optional user input */
      3.2 For i = 0 to 1 do
          a. XVAL = (XKEY + XSEED_j) mod 2^b
          b. w_i = G(t, XVAL)
          c. XKEY = (1 + XKEY + w_i) mod 2^b
      3.3 x_j = w_0|w_1

Intellectual Property Statement

Full Copyright Statement

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

Acknowledgment

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