3GPP TSG SA WG3 Security — S3#20 **S3-010603**

27 - 30 November, 2001

Sophia Antipolis, France

Agenda Item:

Source: Ericsson, Nokia

Title: EAP extension drafts – new versions

Document for: Informative

1 Scope and objectives

Ericsson and Nokia has submitted new versions of EAP related Internet Drafts to IETF:

- Regarding AKA within EAP, the new version (draft-arkko-pppext-eap-aka-01.txt) has been modified according to some requests in the IETF mailing list, e.g. the exact way used to signal that GSM authentication is not acceptable has been changed. Also, a new (optional) feature allowing identity privacy in a manner similar to EAP SRP has been added.
- Regarding EAP within HTTP/SIP, the new version (draft-torvinen-http-eap-01.txt) has gone through some editorial modifications. More precise rules for the use of realm parameter and quotation marks have been added.

Both Internet Drafts will be discussed in the next IETF meeting (Salt Lake City, December 9-14, 2001).

J. Arkko
Ericsson
H. Haverinen
Nokia

Internet Draft
Document: draft-arkko-pppext-eap-aka-01.txt
Expires: December 2001

November 2001

EAP AKA Authentication

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html.

Abstract

This document specifies an Extensible Authentication Protocol (EAP) mechanism for authentication and session key distribution using the UMTS AKA authentication mechanism. AKA is based on symmetric keys, and runs typically in a UMTS Subscriber Identity Module, a smart card like device. AKA provides also backward compatibility to GSM authentication, making it possible to use EAP AKA for authenticating both GSM and UMTS subscribers.

Table of Contents

Status of this Memo.....1
Abstract.....1
1. Introduction and Motivation.....2
2. Conventions used in this document.....3
3. Protocol Overview.....5
4. IMSI Privacy Support.....10
5. Message Format.....12

6. Messages.....13
6.1. EAP-Response/Identity.....13
6.2. EAP-Request/USIM-Challenge.....14
6.3. EAP-Response/USIM-Challenge.....18
6.4. EAP-Response/USIM-Authentication-Reject.....20
6.5. EAP-Response/USIM-Synchronization-Failure.....20
6.6. EAP-Request/USIM-IMSI.....21
6.7. EAP-Response/USIM-IMSI.....22
7. Interoperability with GSM.....23
8. IANA and Protocol Numbering Considerations.....24
9. Security Considerations.....24
10. Intellectual Property Right Notices.....24
Acknowledgements.....25
Authors' Addresses.....25

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 [1]. The Universal Mobile Telecommunications System (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), a smart card like device. However, the applicability of AKA is not limited to client devices with smart cards, but the AKA mechanisms could also be implemented in host software, for example AKA also provides backward compatibility to the GSM authentication mechanism [2]. Compared to the GSM mechanism, AKA provides substantially longer key lengths and the authentication of the server side as well as the client side.

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 and IEEE 801.1x technology through EAP over Wireless [3, 4].

- 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 reference [1].

It is also possible that the home environment delegates the actual authentication task to an intermediate node. In this case the authentication vector or parts of it are delivered to the intermediate node, enabling it to perform the comparison between RES and XRES, and possibly also use CK and IK. In EAP AKA, the EAP server node is such an intermediate node.

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) [5].

2. Conventions used in this document

The following terms will be used through this document:

AAA protocol

Authentication, Authorization and Accounting protocol

AAA server

In this document, AAA server refers to the network element that resides on the border of Internet AAA network and GSM network. Cf. EAP server

AKA

Authentication and Key Agreement

AuC

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

EAP

Extensible Authentication Protocol [6].

EAP server

The network element that terminates the EAP protocol. Typically, the EAP server functionality is implemented in a AAA server.

GSM

Global System for Mobile communications.

NAI

Network Access Identifier [5].

AUTN

Authentication value generated by the AuC which together with the RAND authenticates the server to the client, 128 bits [1].

AUTS

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

RAND

Random number generated by the AuC, 128 bits [1].

RES

Authentication result from the client, which together with the RAND authenticates the client to the server, 128 bits [1].

SQN

Sequence number used in the authentication process, 48 bits [1].

SIM

Subscriber Identity Module. SIM cards are smart cards distributed by GSM operators.

SRES

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

USIM

UMTS Subscriber Identity Module. These cards are smart cards Similar to SIMs and are distributed by UMTS operators.

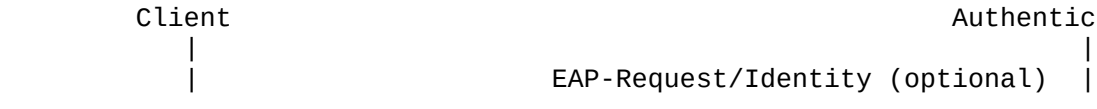
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [7]

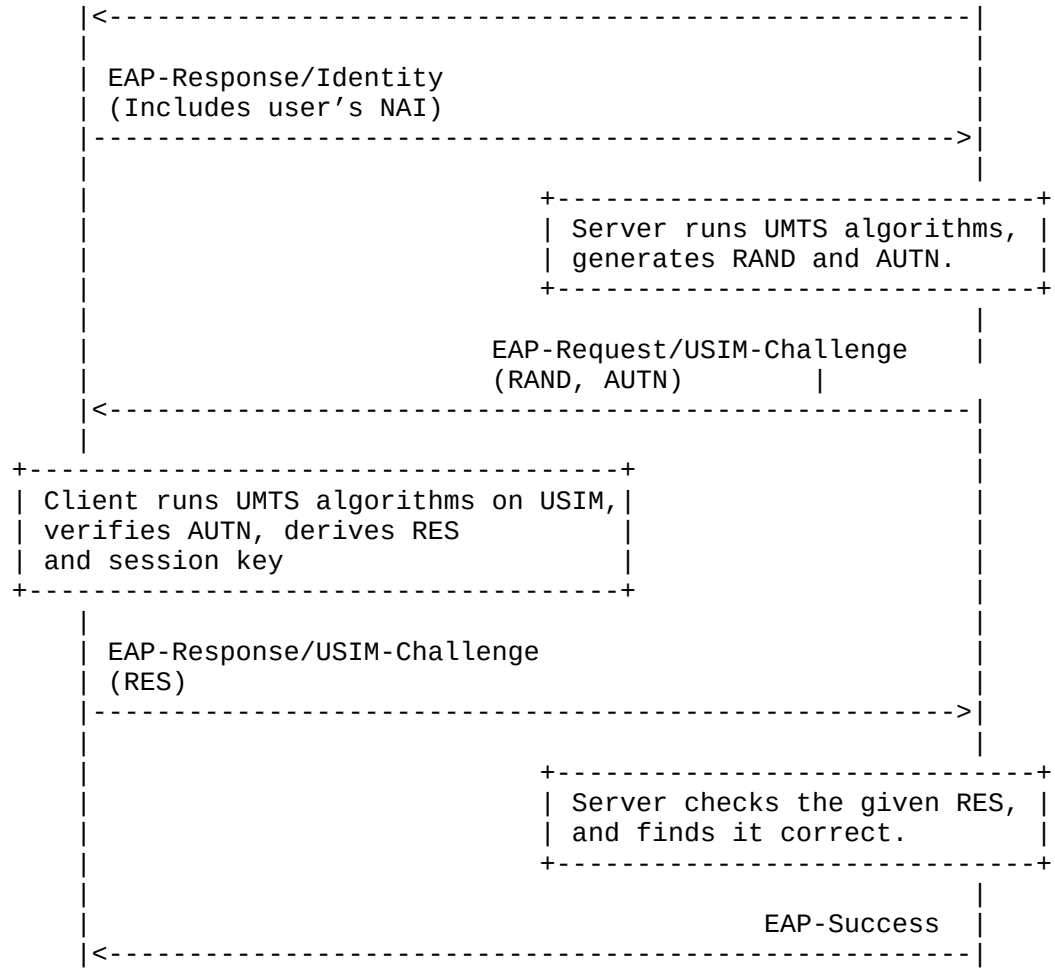
3. Protocol Overview

In this document, the term EAP Server refers to the network element that terminates the EAP protocol. Usually the EAP server is separate from the authenticator device, which is the network element closest to the client, such as a Network Access Server (NAS) or an IEEE 802.1X bridge. Typically, the authenticator does not contain the EAP server functionality, but the EAP server functionality is implemented on a separate AAA server with whom the authenticator communicates using an AAA protocol. (The exact AAA communications is outside the scope of this document, however.)

The below message flow shows the basic successful authentication case with the EAP AKA. The EAP AKA uses two roundtrips to authorize the user and generate session keys. As in other EAP schemes, first an identity request/response message pair is exchanged. (For this particular EAP protocol, the identity request is defined to be optional, to shorten the authentication process to a minimal one.)

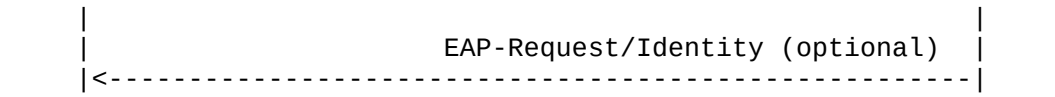
Next, the EAP server starts the actual AKA protocol by sending an EAP-Request/USIM-Challenge message. This message contains a random number (RAND) and an authorization vector (AUTN). The EAP-Request/USIM-Challenge message MAY optionally contain encrypted data, which is used for IMSI privacy support, as described in Section 4. The encrypted data is not shown in the figures of this section. The client runs the AKA algorithm (perhaps inside an USIM) and verifies the AUTN. If this is successful, the client is talking to a legitimate EAP server and proceeds to send the EAP-Response/USIM-Challenge. This message contains a result parameter that allows the EAP server in turn to verify that the client is a legitimate one.

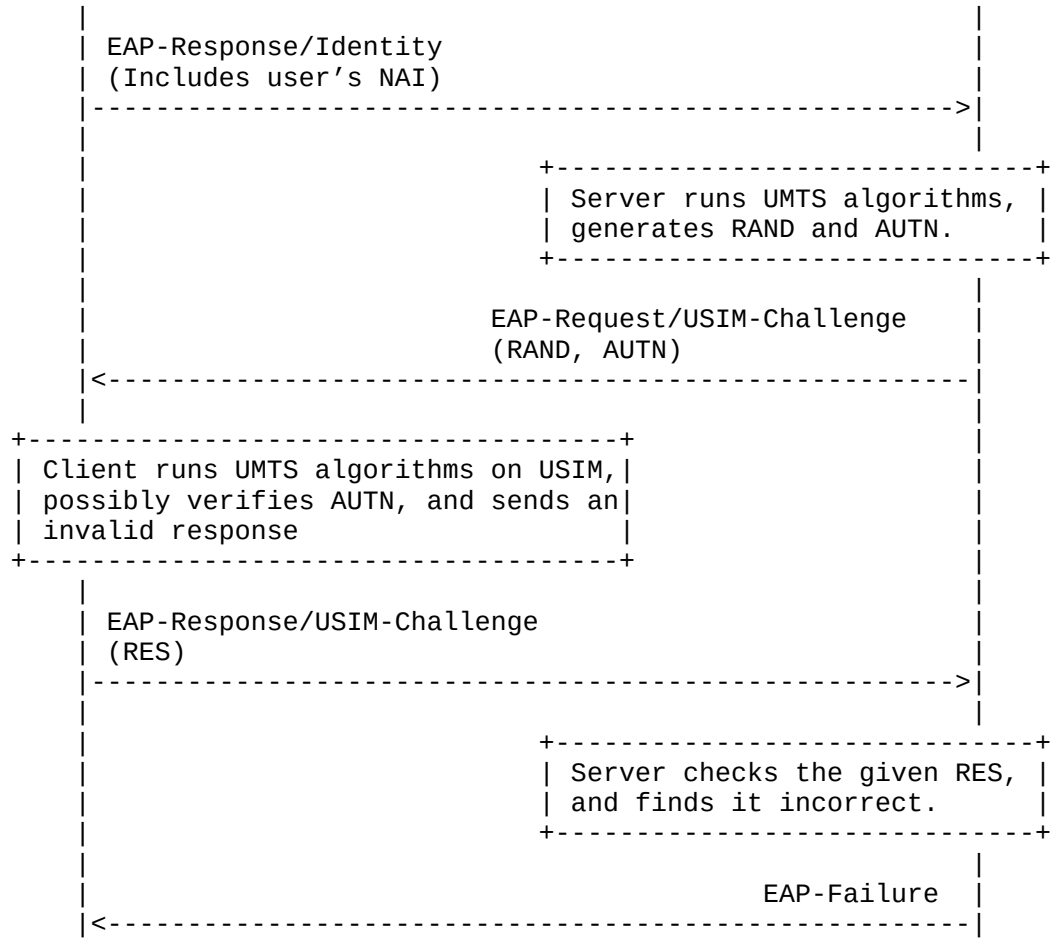




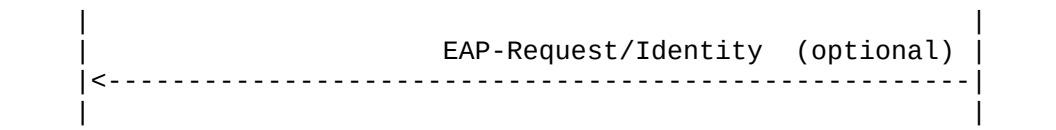
When EAP AKA is run in the GSM compatible mode, the message flow is otherwise identical to the message flow below except that the AUTN attribute is not included in EAP-Request/USIM-Challenge packet.

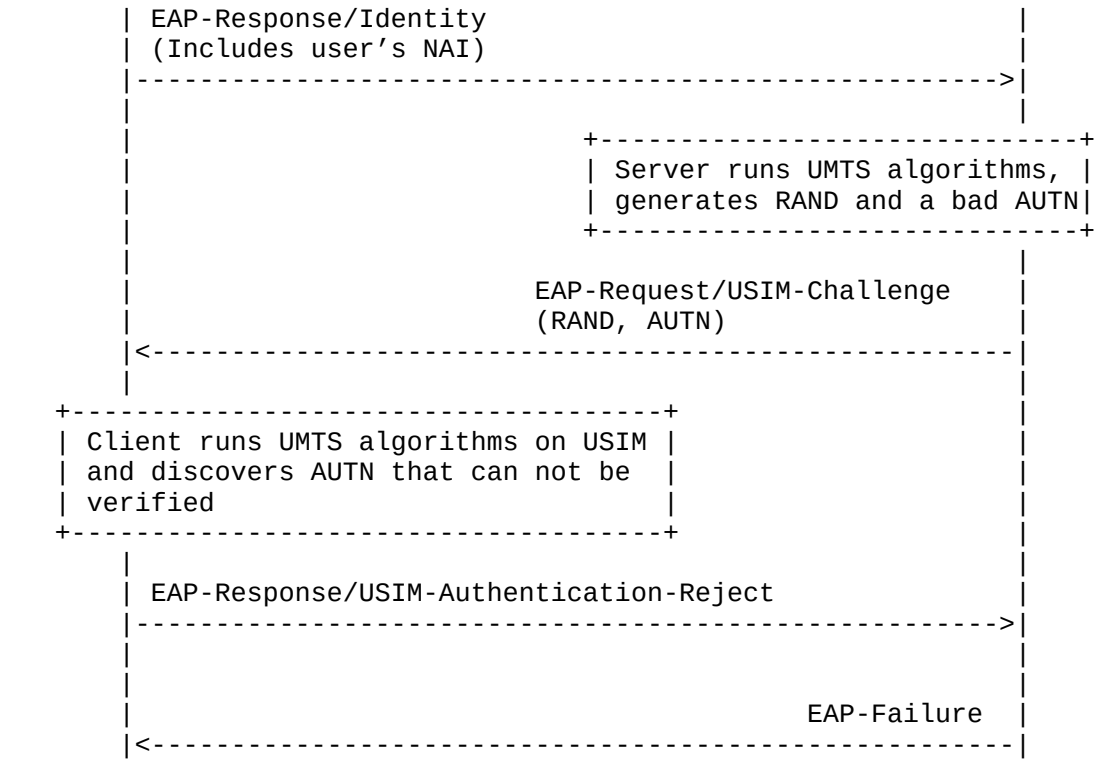
The second message flow shows how the EAP server rejects the Client due to failed authentication. The same flow is also used in the GSM compatible mode, except that the AUTN parameter is not included in the EAP-Request/USIM-Challenge packet.



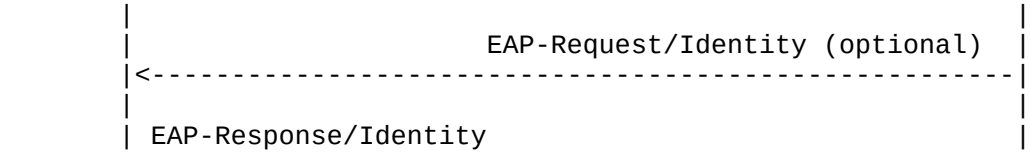


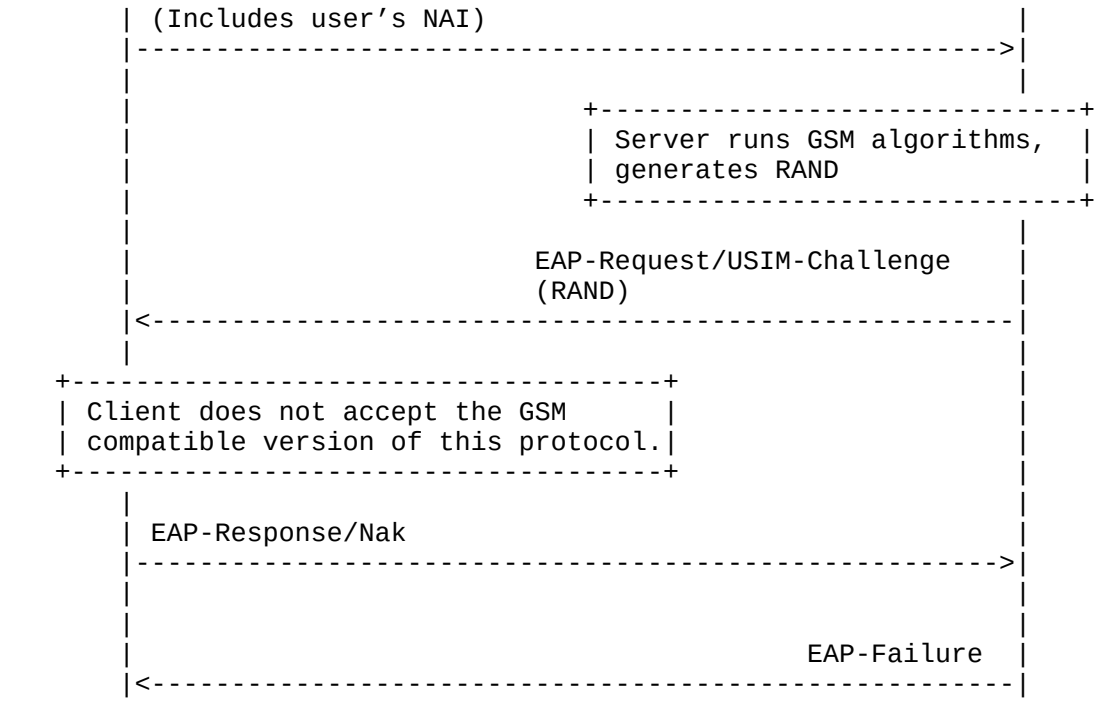
The next message flow shows the client rejecting the AUTN of the EAP server. This flow is not used in the GSM compatible mode.



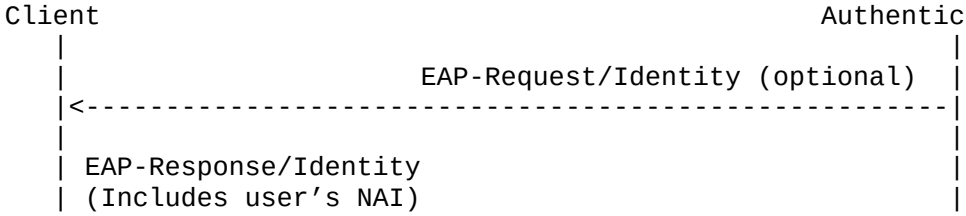


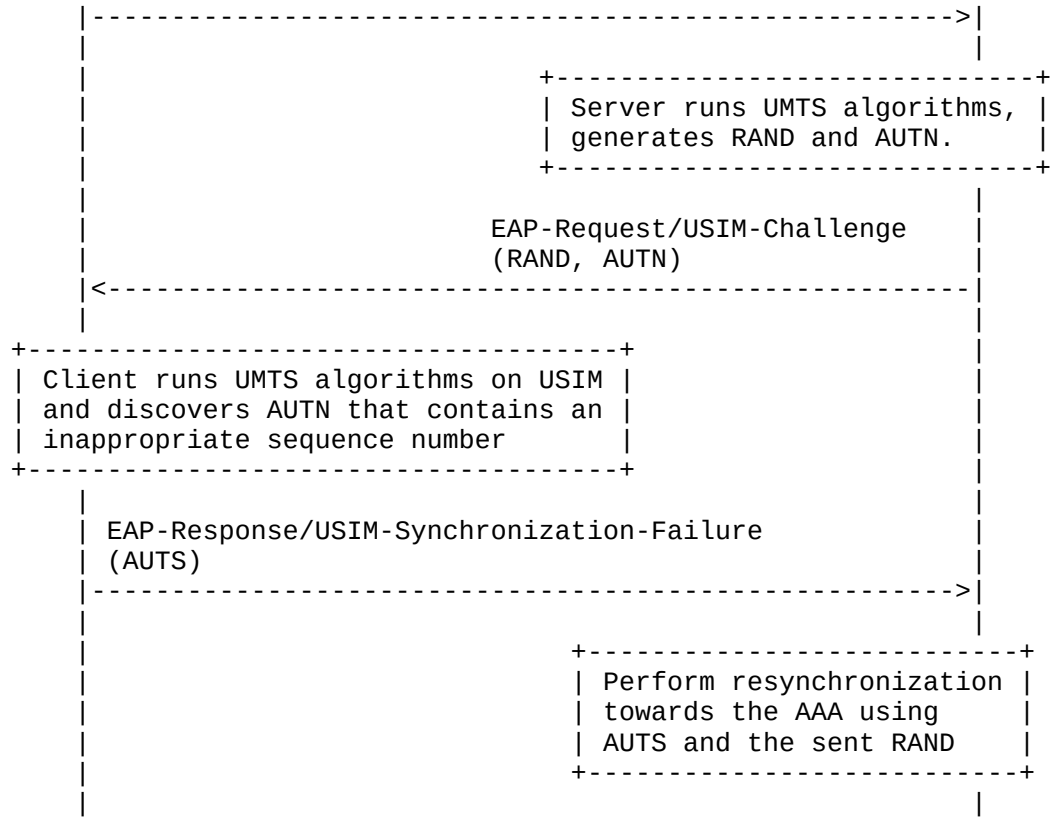
Networks that are not UMTS aware use the GSM compatible version of this protocol even for UMTS subscribers. In this case, the AUTN parameter is not included in the EAP-Request/USIM-Challenge packet. If a UMTS capable client does not want to accept the use of the GSM compatible mode, the client can reject the authentication with the EAP-Response/Nak message [6], as shown in the following figure:





The AKA uses shared secrets between the Client and the Client’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. Here’s what happens then:





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

4. IMSI Privacy Support

In the very first connection to an EAP server, the client always transmits the cleartext IMSI in the EAP-Response/Identity packet. In subsequent connections, the optional IMSI privacy support can be used to hide the IMSI and to make the connections unlinkable to a passive eavesdropper.

The EAP-Request/USIM-Challenge message MAY include an encrypted pseudonym in the value field of the AT_ENCR_DATA attribute. The AT_IV and AT_MAC attributes are also used to transport the pseudonym to the client, as described in Section 6.2. Because the IMSI privacy support is optional to implement, the client MAY ignore the AT_IV, AT_ENCR_DATA, and AT_MAC attributes and always transmit the IMSI in the EAP-Response/Identity packet.

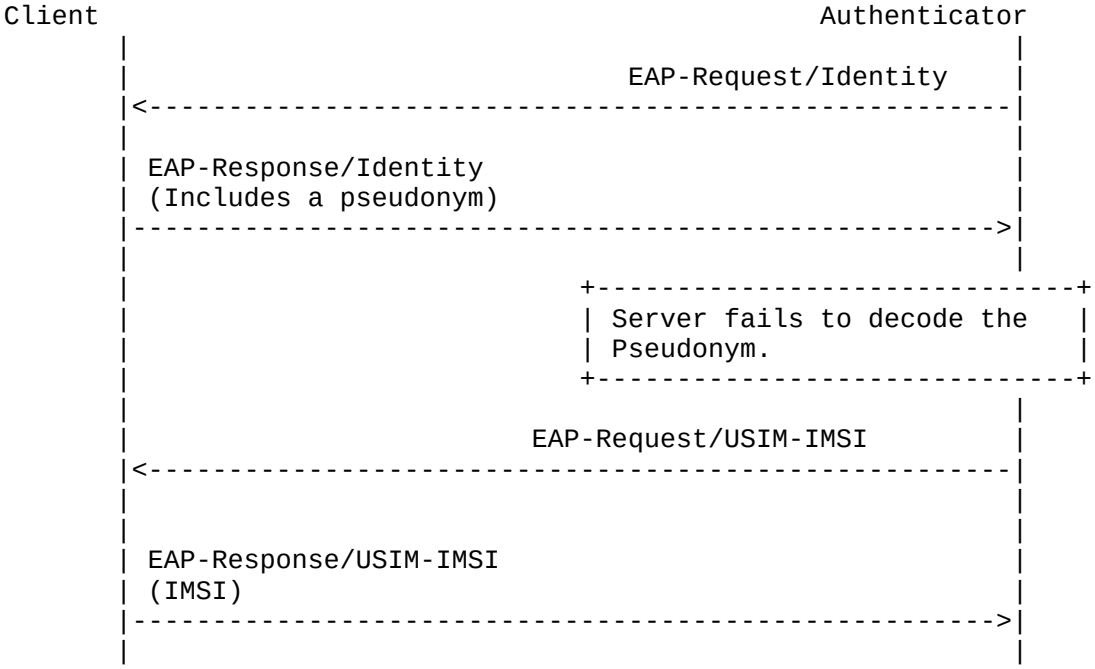
On receipt of the EAP-Request/USIM-Challenge, the client verifies the AT_AUTN attribute before looking at the AT_ENCR_DATA or AT_MAC attributes. If the AUTN is invalid, then the client MUST ignore the AT_IV, AT_ENCR_DATA and AT_MAC attributes. If AUTN is valid, then the client MAY derive the K_encr and K_int keys as described in Section 6.2 and verify the AT_MAC attribute. If the AT_MAC attribute is valid, then the client MAY decrypt the encrypted data and use the pseudonym in the next authentication. If the MAC is invalid, then

the encrypted data MUST be ignored and the whole EAP packet MAY be silently ignored.

The EAP server produces pseudonyms in an implementation-dependent manner. Please see [8] for examples on how to produce pseudonyms. The pseudonyms need to be reversible to the IMSI only on the EAP server. Regardless of construction method, the pseudonym MUST conform to the grammar specified for the username portion of an NAI.

On the next connection to the EAP server, the client MAY transmit the received pseudonym in the first EAP-Response/Identity packet. The client concatenates the received pseudonym with the "@" character and the NAI realm portion. The client MUST use the same realm portion that it used in the connection when it received the pseudonym.

If the EAP server fails to decode the pseudonym to a known client name, then the EAP server requests the regular IMSI (non-pseudonym identity) by issuing the EAP-Request/USIM-IMSI packet to the client. This packet includes no attributes. The client responds with the EAP-Response/USIM-IMSI, which includes the client's IMSI in the clear. This case is illustrated in the figure below.

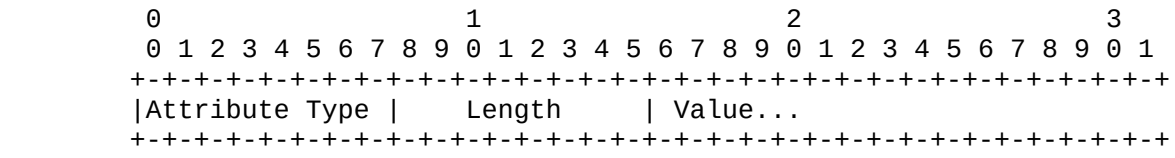


After receiving the EAP-Response/USIM-IMSI packet, the EAP server issues the EAP-Request/USIM-Challenge and the authentication proceeds as usual.

Because the keys that are used to protect the pseudonym are derived from the AKA cipher key (CK) and the AKA integrity key (IK), the IMSI privacy support is not available when EAP AKA is used in the GSM compatible mode.

5. Message Format

The Type-Data of the EAP AKA packets begins with a 1-octet Subtype field, which is followed by a 2-octet reserved field. The rest of the Type-Data 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 may be two or more bytes in length. The type and length fields determine the format and length of the value field.

When an attribute numbered within the range 0 through 127 is encountered but not recognized, the EAP/USIM message containing that attribute MUST be silently discarded. These attributes are called non-skippable attributes.

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 in 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.

6. Messages

6.1. EAP-Response/Identity

In the beginning of EAP authentication, the Authenticator issues the EAP-Request/Identity packet to the client. The client responds with EAP-Response/Identity, which contains the user's identity. The formats of these packets are specified in [6].

The EAP AKA mechanism uses the NAI format [5] as the identity. In order to facilitate the use of the existing cellular roaming infrastructure, the subscriber's IMSI is used as the client identifier. When IMSI privacy is not used, the EAP AKA client transmits the user's IMSI within the NAI in the EAP Response/Identity packet. The NAI is of the format "0imsi@realm". In other words, the first character is the digit zero (ASCII value 0x30), followed by the IMSI, followed by the @ character and the realm. The IMSI is an ASCII string that consists of not more than 15 decimal digits (ASCII values between 0x30 and 0x39) as specified in [9].

When the optional IMSI privacy support is used, the client MAY use the pseudonym received as part of the previous authentication sequence as the user name portion of the NAI, as specified in Section 4.

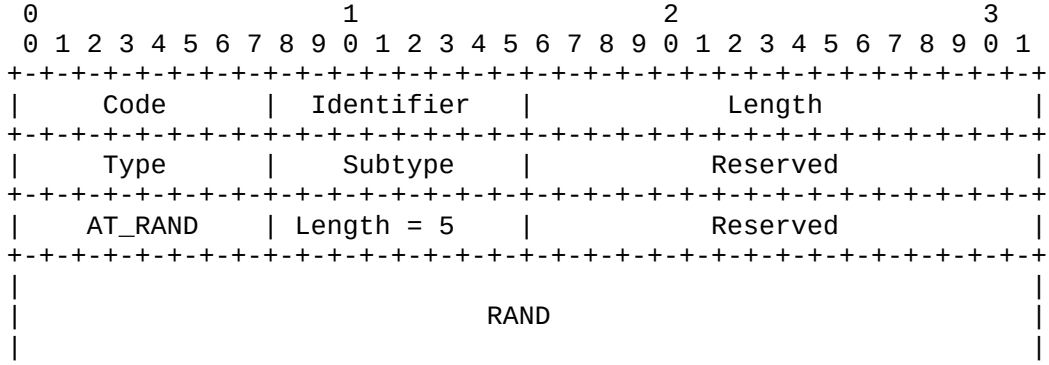
The AAA network routes AAA requests to the correct AAA server using the realm part of the NAI. Because cellular roaming can be used with EAP AKA, the AAA request can be routed to an AAA server in the visited network instead of the server indicated in the NAI realm. The operators need to agree on this special AAA routing in advance. It is recommended that operators should reserve the realm portion of NAI for EAP AKA users exclusively, so that exactly the same realm is not used with other authentication methods. This convention makes it easy to recognize that the NAI identifies a UMTS or GSM subscriber of this operator, which may be useful when configuring the routing rules in the visited AAA networks.

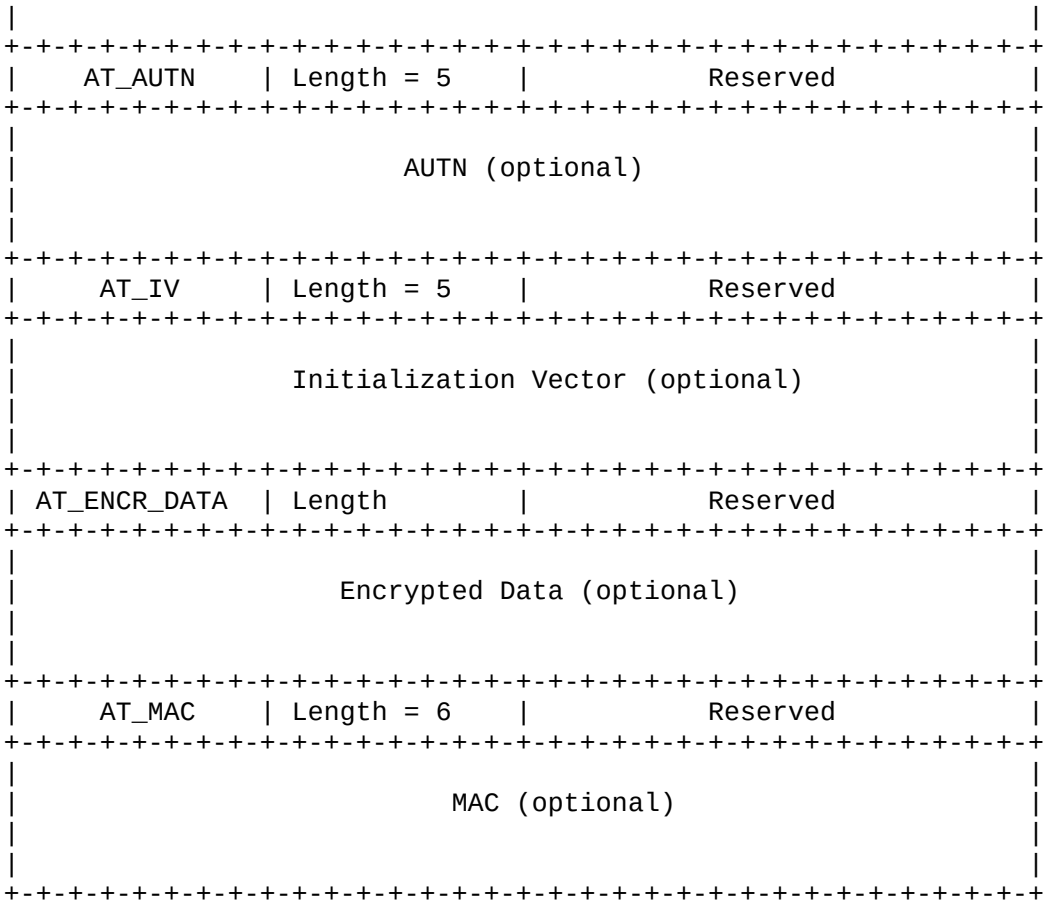
In the EAP AKA protocol, the EAP-Request/Identity message is optional when applicable. If the client can positively determine that it has to authenticate, it MAY send an unsolicited EAP-Response/Identity to the authenticator with an EAP Identifier value it has picked up itself. The client MUST NOT send an unsolicited EAP-Response/Identity if it has already received an EAP-Request/Identity packet. The client MUST send an EAP-Response/Identity to all received EAP-Request/Identity packets,

using the Identifier value in the EAP-Request/Identity. If the authenticator receives an unsolicited EAP-Response/Identity, it SHOULD process the packet as if it had requested it. If the authenticator receives an EAP-Response/Identity with an incorrect Identifier value in response to the first EAP-Request/Identity it has sent to the client, then the authenticator SHOULD still accept the EAP-Response/Identity packet.

6.2. EAP-Request/USIM-Challenge

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


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Code
Identifier
Length
Type
Subtype
Reserved
AT_RAND
Length = 5
Reserved
RAND



The semantics of the fields is described below:

Code

   1 for Request

Identifier

   See [6]

Length

   The length of the EAP Request packet.

Type

   TBD

Subtype

   1 for USIM-Challenge

Reserved

Set to zero when sending, ignored on reception.

AT_RAND

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. The AT_RAND attribute MUST be present in EAP-Request/USIM-Challenge.

AT_AUTN

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. The AT_AUTN attribute MUST NOT be included in the GSM compatible mode of this protocol; otherwise it MUST be included.

AT_IV

The value field 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. This attribute MUST be included if and only if the AT_ENCR_DATA is included. Messages that do not meet this condition MUST be silently discarded.

AT_ENCR_DATA

The AT_ENCR_DATA MAY is optional. The value field of this attribute consists of two reserved bytes followed by bytes encrypted using the Advanced Encryption Standard (AES) [10] 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 [11] for a description of the CBC mode.

The encryption key (K_encr) is derived from the AKA Cipher Key (CK) with the following formula. The result of the SHA-1 hash value [12] is truncated to 128 bits by ignoring the 32 rightmost

bits. The notation A|0 denotes A concatenated with the byte zero 0x00.

K_encr = 128 leftmost bits of SHA1(CK|0)

The plaintext consists of nested attributes as described below.

AT_MAC

This attribute is optional, but it MUST be included whenever the AT_ENCR_DATA attribute is included. Messages that do not meet this condition MUST be silently discarded.

The value field of the AT_MAC attribute contains two reserved

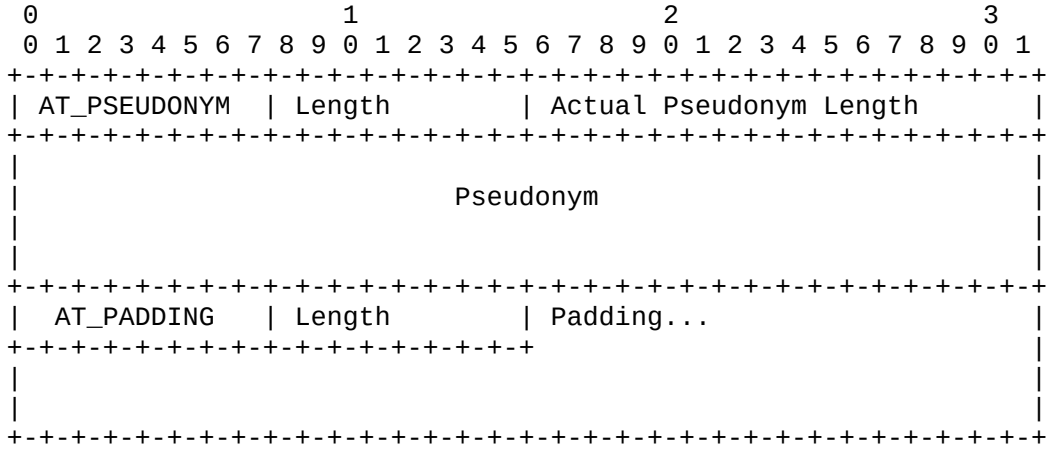
bytes followed by a message authentication code (MAC). The MAC is calculated over the whole EAP packet with the exception that the value field of the MAC attribute is set to zero when calculating the MAC. The reserved bytes are set to zero when sending and ignored on reception.

The MAC algorithm is HMAC-SHA1 [13] keyed hash value, so the length of the MAC is 20 bytes.

The integrity protection key (K_int) used in the calculation of the MAC is derived from the AKA integrity key (IK) with the following formula. The notation A|0 denotes A concatenated with the byte zero 0x00.

```
K_int = SHA1(IK|0)
```

The AT_IV, AT_ENCR_DATA and AT_MAC attributes are used for IMSI privacy. The plaintext of the AT_ENCR_DATA value field consists of nested attributes, which are shown below. Later versions of this protocol MAY specify additional attributes to be included within the encrypted data.



AT_PSEUDONYM

This attribute is optional. The value field of this attribute begins with 2-byte actual pseudonym length, which specifies the length of the pseudonym in bytes. This field is followed by a pseudonym user name, of the indicated actual length, that the client can use in the next authentication, as described in Section 4. The user name 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.

AT_PADDING

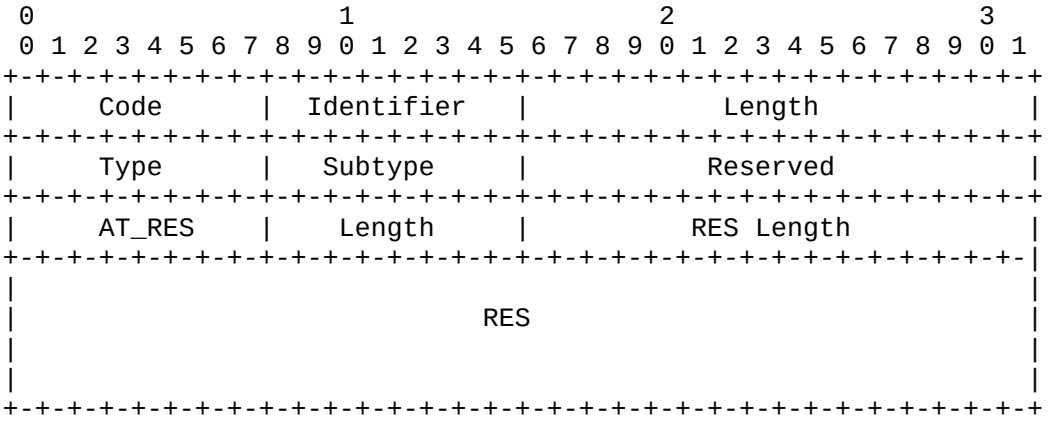
The encryption algorithm requires the length of the plaintext to

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 (0x00) on sending. The recipient of the message MUST verify that the pad bytes are set to zero, and silently drop the message if this verification fails.

6.3. EAP-Response/USIM-Challenge

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

EAP-Response/USIM-Challenge MAY include the AT_MAC attribute to integrity protect the EAP packet. 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. AT_MAC, AT_ENCR_DATA and AT_IV attributes are not shown in the figure below. If present, they are processed as in EAP-Request/USIM-Challenge packet. The EAP server MUST process EAP-Response/USIM-Challenge messages that include these attributes even if the server did not implement these optional attributes.



The semantics of the fields is described below:

Code

2 for Response

Identifier

See [6]

Length

The length of the EAP Response packet.

Type

TBD

Subtype

1 for USIM-Challenge

Reserved

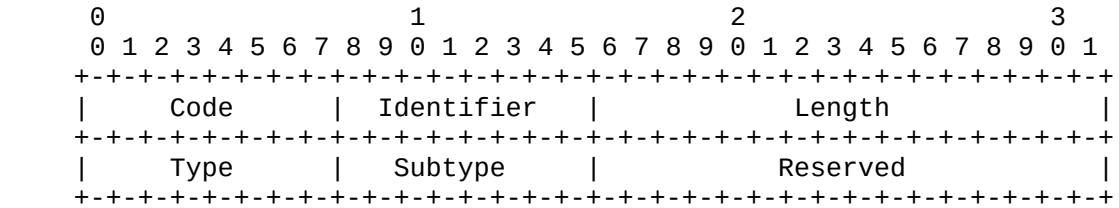
Set to zero when sending, ignored on reception.

AT_RES

This attribute MUST be included in EAP-Response/USIM-Challenge. The value field of this attribute begins with the 2-byte RES Length, which is identifies the exact length of the RES (or SRES) in bits. The RES length is followed by the UMTS AKA RES or GSM SRES parameter. According to the specification [14] the length of the AKA RES can vary between 32 and 128 bits. The GSM SRES parameter is always 32 bits long. 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.

6.4. EAP-Response/USIM-Authentication-Reject

The format of the EAP-Response/USIM-Authentication-Reject packet is shown below.



The semantics of the fields is described below:

Code

2 for Response

Identifier

See [6]

Length

The length of the EAP Response packet.

Type

TBD

Subtype

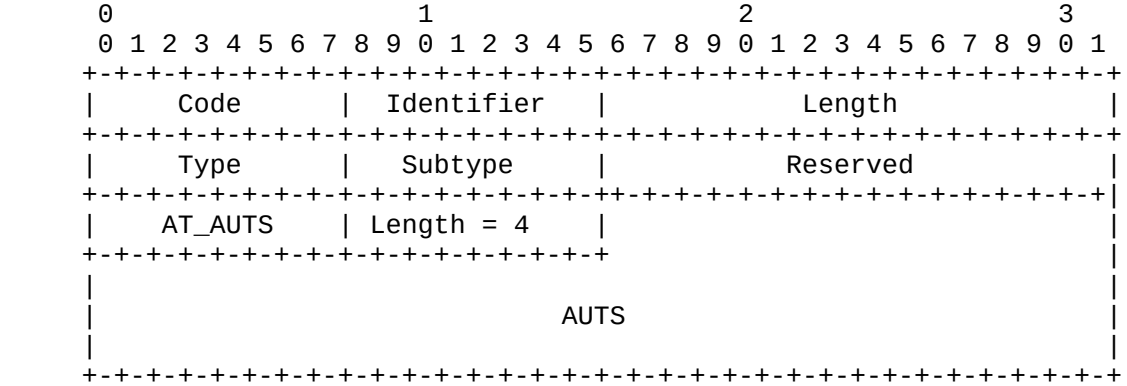
2 for USIM-Authentication-Reject

Reserved

Set to zero on sending, ignored on reception.

6.5. EAP-Response/USIM-Synchronization-Failure

The format of the EAP-Response/USIM-Synchronization-Failure packet is shown below.



The semantics of the fields is described below:

Code

2 for Response

Identifier

See [6]

Length

The length of the EAP Response packet, 20.

Type

TBD

Subtype

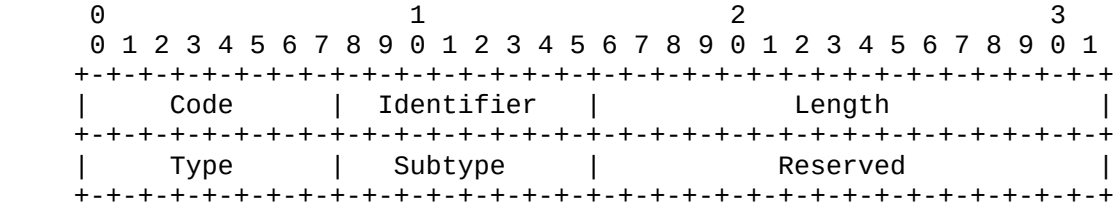
4 for USIM-Synchronization-Failure

AT_AUTS

This attribute MUST be included in EAP-Response/USIM-Synchronization-Failure. The value field of this attribute contains the AKA AUTS parameter, 112 bits (14 bytes).

6.6. EAP-Request/USIM-IMSI

The format of the EAP-Request/USIM-IMSI packet is shown below.



The semantics of the fields is described below:

Code

1 for Request

Identifier

See [6]

Length

The length of the EAP Request packet.

Type

TBD

Subtype

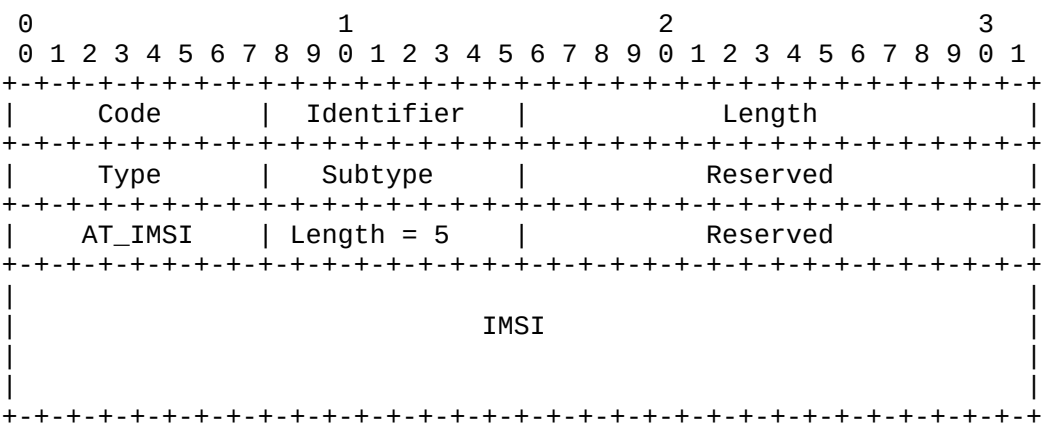
5 for USIM-IMSI

Reserved

Set to zero on sending, ignored on reception.

6.7. EAP-Response/USIM-IMSI

The format of the EAP-Response/USIM-IMSI packet is shown below.



The semantics of the fields is described below:

Code

2 for Response

Identifier

See [6]

Length

The length of the EAP Response packet.

Type

TBD

Subtype

5 for USIM-IMSI

Reserved

Set to zero on sending, ignored on reception.

AT_IMSI

This attribute MUST be included in EAP-Response/USIM-IMSI. The value field of this attribute contains two reserved bytes followed by the IMSI, represented as an ASCII string that consists of not more than 15 decimal digits (ASCII values between 0x30 and 0x39) [9]. The reserved bytes are set to zero on sending and ignored on reception. The IMSI characters are followed by one or more "F" characters (ASCII value 0x46). They are included to make the length of the value field 16 bytes.

7. Interoperability with GSM

The EAP AKA protocol is able to authenticate both UMTS and GSM users, if the subscriber's operator's network is UMTS aware. This is because the home network will be able to determine from the subscriber records whether the subscriber is equipped with a UMTS USIM or a GSM SIM. A UMTS aware home network will hence always use UMTS AKA with UMTS subscribers and GSM authentication with GSM subscribers. With GSM subscribers, the EAP AKA protocol is always used in the GSM compatible mode.

It is not possible to use a GSM AuC to authenticate UMTS subscribers. (Note that if the home network doesn't support an authentication method it should not distribute SIMs for that method.)

However, it is possible that the node actually terminating EAP and the node that stores the authentication keys (AuC) are separate, and support different authentication types. If the node terminating EAP is GSM-only but AuC is UMTS-aware, then authentication can still be achieved using the GSM compatible version of EAP AKA. This authentication will be weaker, since the GSM compatible mode does

not provide for mutual authentication. Section 6.8.1.1 in [1] specifies how the GSM SRES parameter and the Kc key can be calculated on the USIM and the AuC. If a UMTS terminal does not want to accept the GSM compatible version of this protocol, then it can reject the authentication with the EAP-Response/USIM-GSM-Authentication-Reject packet.

In conclusion, the following table shows which variant of the EAP AKA protocol should be run under different conditions:

SIM	EAP node	AuC	EAP AKA mode
GSM	(any)	(any)	GSM
UMTS	(any)	GSM	(illegal)
UMTS	GSM	GSM+UMTS	GSM
UMTS	GSM+UMTS	GSM+UMTS	UMTS

8. IANA and Protocol Numbering Considerations

IANA has assigned the number TBD for EAP AKA authentication.

EAP AKA messages include a Subtype field. The following Subtypes are specified:

```
USIM-Challenge.................................1
USIM-Authentication-Reject.....................2
USIM-Synchronization-Failure...................4
USIM-IMSI......................................5
```

The Subtype-specific data is composed of attributes, which have attribute type numbers. The following attribute types are specified:

```
AT_RAND........................................1
AT_AUTN........................................2
AT_RES.........................................3
AT_AUTS........................................4
AT_IMSI........................................5
AT_PADDING.....................................6

AT_IV.........................................129
AT_ENCR_DATA..................................130
AT_MAC........................................131
AT_PSEUDONYM..................................132
```

9. Security Considerations

Implementations running the EAP AKA protocol will rely on the security of the AKA scheme, and the secrecy of the symmetric keys stored in the USIM and the AuC.

10. Intellectual Property Right Notices

On IPR related issues, Nokia and Ericsson refer to the their respective statements on patent licensing. Please see http://www.ietf.org/ietf/IPR/NOKIA and http://www.ietf.org/ietf/IPR/ERICSSON-General

Acknowledgements

The authors wish to thank Rolf Blom of Ericsson, Bernard Aboba of Microsoft, Arne Norefors of Ericsson, N.Asokan of Nokia and Jukka-Pekka Honkanen of Nokia for interesting discussions in this problem space.

The IMSI privacy support is based on the identity privacy support of [8]. The attribute format is based on the extension format of Mobile IPv4 [15].

Authors' Addresses

Jari Arkko
Ericsson
02420 Jorvas
Finland

Phone: +358 40 5079256
Email: jari.arkko@ericsson.com

Henry Haverinen
Nokia Mobile Phones
P.O. Box 88
33721 Tampere
Finland
Phone: +358 50 594 4899
E-mail: henry.haverinen@nokia.com

References

[1] 3GPP Technical Specification 3GPP TS 33.102 V3.6.0: "Technical Specification Group Services and System Aspects; 3G Security; Security Architecture (Release 1999)", 3rd Generation Partnership Project, November 2000.

[2] GSM Technical Specification GSM 03.20 (ETS 300 534): "Digital cellular telecommunication system (Phase 2); Security related network functions", European Telecommunications Standards, Institute, August 1997.

[3] IEEE P802.1X/D11, "Standards for Local Area and Metropolitan Area Networks: Standard for Port Based Network Access Control", March 2001

[4] IEEE Draft 802.11eS/D1, "Draft Supplement to STANDARD FOR Telecommunications and Information Exchange between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: Specification for Enhanced Security", March 2001

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

[6] L. Blunk, J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998.

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

[8] J. Carlson, B. Aboba, H. Haverinen, "EAP SRP-SHA1 Authentication Protocol", draft-ietf-pppext-eap-srp-03.txt, July 2001 (work-in-progress)

[9] GSM Technical Specification GSM 03.03 (ETS 300 523): "Digital cellular telecommunication system (Phase 2); Numbering, addressing and identification", European Telecommunications Standards Institute, April 1997.

[10] Federal Information Processing Standard (FIPS) draft standard, "Advanced Encryption Standard (AES)", http://csrc.nist.gov/publications/drafts/dfips-AES.pdf, September 2001

[11] US National Bureau of Standards, "DES Modes of Operation",

Federal Information Processing Standard (FIPS) Publication 81, December 1980.

[12] Federal Information Processing Standard (FIPS) Publication 180-1, "Secure Hash Standard," National Institute of Standards and Technology, U.S. Department of Commerce, April 17, 1995.

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

[14] 3GPP Technical Specification 3GPP TS 33.105 V3.5.0: "Technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements (Release 1999)", 3rdGeneration Partnership Project, October 2000

[15] C. Perkins (editor), "IP Mobility Support", RFC 2002, October 1996

INTERNET-DRAFT
Document: draft-torvinen-http-eap-01.txt
Expires: May 2002

J. Arkko
V. Torvinen
Ericsson
A. Niemi
Nokia

November 2001

HTTP Authentication with EAP

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html.

Abstract

This document describes a HTTP authentication scheme using PPP Extensible Authentication Protocol (EAP).

HTTP EAP authentication enables HTTP connections to be authenticated using any of the authentication schemes supported through EAP. EAP performs the authentication without sending the password in the clear text format (which is the biggest weakness of the Basic HTTP authentication scheme, for example).

It is useful for HTTP protocol because it opens up several new authentication schemes without additional specification work. The same benefits can be reached by any other protocols, which apply HTTP authentication, such as Session Initiation Protocol (SIP).

Table of Contents

1 Introduction.....2

2 HTTP EAP Authentication Scheme.....2

2.1 The WWW-Authenticate Response Header.....4
2.2 The Authorization Request Header.....6
2.3 Authentication-Info Response Header.....6

3 Security Considerations.....7

4 References.....9

5 Acknowledgements.....9

6 Author's Addresses.....10

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 RFC-2119 [1]

1 Introduction

The HTTP Authentication framework includes two authentication schemes: Basic and Digest [2]. In the Basic scheme, the client authenticates itself with a user-ID and a password for each realm. The Basic scheme is perceived as insecure since the user credentials are transmitted across the public network in a cleartext format. The Digest scheme is based on cryptographic hashes and is consequently perceived as a more secure authentication scheme than Basic, but is limited the use of passwords. See [2] for detailed information about the general HTTP authentication protocol.

The PPP Extensible Authentication Protocol (EAP) is a general protocol for PPP authentication [3]. Even though EAP was originally developed as a link layer protocol, it can also be applied at the application layer. EAP supports multiple authentication mechanism (e.g. smart cards, Kerberos, Public Key, One Time Passwords, and others) and it can, by definition, be easily extended to support new authentication mechanisms [see e.g. 4, 5, 6, 7]. EAP packets are defined in a binary format, and their contents depend highly on the used authentication scheme.

HTTP EAP Authentication Scheme supplements HTTP Authentication with EAP functionality. This opens up several new authentication schemes for HTTP Authentication without additional specification work.

2 HTTP EAP Authentication Scheme

The HTTP EAP Authentication Scheme delivers base64 encoded EAP packets within HTTP Authentication headers (e.g. WWW-Authenticate Response headers and Authorization Request headers). EAP packets include all relevant information about the required authentication scheme, e.g. authentication scheme, packet type (request, response,

success or failure) and/or challenge. The content of these packets is up to the chosen EAP authentication scheme.

The progression of an authentication procedure depends also on the chosen authentication mechanism. Typically, the authenticator sends an initial Identity Request followed by one or more Requests for authentication information. The peer sends a Response packet in reply to each Request. As with the Request packet, the Response packet contains a type field, which corresponds to the type field of the Request. The authenticator ends the authentication phase with a Success or Failure packet. See Figure 1.

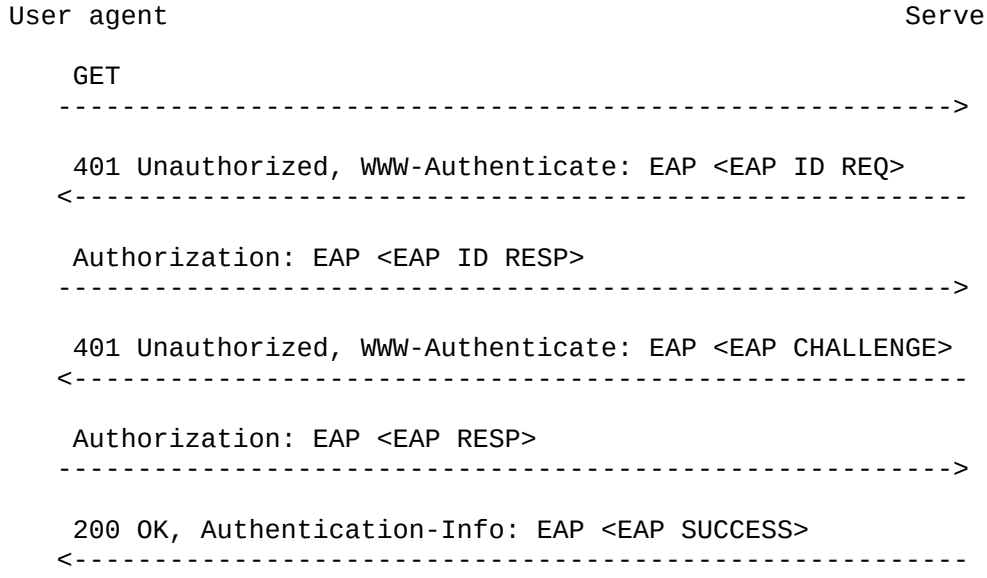


Figure 1. HTTP EAP Authentication message flow

This message flow above represents only the typical situation. Variations of the flow are also possible in the following situations:

- The chosen authentication mechanism requires more than the single challenge-response message pair shown. Any number of message exchanges are allowed here.
- Error situations result in terminating the flow from the server's side with an error response. This response could be one of 401 Unauthorized, 403 Forbidden, or 407 Proxy Authentication Required. For 401 and 407, the client distinguishes the error situation from the continuation of the EAP exchange by the existence of EAP FAILURE payload, or the lack of any EAP payload.
- Error situations from the client's side result in terminating the communications with the server.
- Certain EAP authentication mechanisms such as [7] allow an optimized flow where identity request does not need to be sent. In

these cases, if the client knows it will be demanded EAP authentication, it can include an unsolicited EAP ID RESP already

in the GET message. This would enable the server to start the actual authentication exchange immediately.
- EAP authentication was shown to be run towards the server which responds with 401 Unauthorized responses. It is also possible to run towards a proxy, which responds with 407 Proxy Authentication Required responses.

In this document, we define three new header types for the HTTP authentication framework. These headers, WWW-Authenticate Response Header, Authorization Request Header and Authentication-Info Response Header, are needed for making EAP as an independent HTTP authentication scheme.

2.1 The WWW-Authenticate Response Header

The general HTTP authentication framework uses an extensible, case-insensitive token to identify the authentication scheme. Authentication scheme identifier is followed by a comma-separated list of attribute-value pairs, which carry the parameters necessary for achieving authentication via that scheme.

```
auth-scheme     = token
auth-param      = token "=" ( token | quoted-string )
```

If a server receives a request for an access-protected object without an acceptable Authorization header, the server responds with a "401 Unauthorized" status code, a WWW-Authenticate header and at least one challenge applicable to the requested resource. A Proxy acts in the same way but it uses a "407 Proxy Authentication Required" status code instead.

```
challenge       = auth-scheme 1*SP 1#auth-param
```

The authentication parameter realm is defined for all authentication schemes:

```
realm           = "realm" "=" realm-value
realm-value     = quoted-string
```

The realm value and the canonical root URL of the server being accessed define the protection space.

The realm directive (case-insensitive) is required for all authentication schemes that issue a challenge. The realm value (case-sensitive) is a string, which may have additional semantics specific to the authentication scheme.

For HTTP EAP Authentication, the framework above is utilized as follows:

```
challenge       = "Eap" eap-challenge
```

```
      eap-challenge   = 1#(realm | eap-param)
```

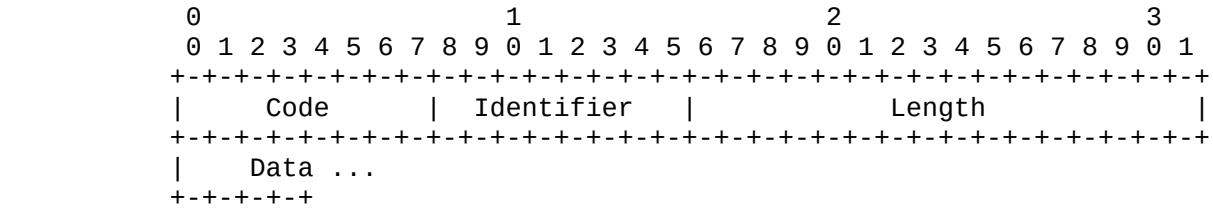


```
      realm           = "realm" "=" <"> realm-value <">
      realm-value     = token [ "@" token ]
      eap-param       = "eap-p" "=" <"> eap-packet <">
      eap-packet      = <base64 encoded eap-packet, except
                         not limited to 76 char/line>
```

The realm value SHOULD be globally unique. Proxy servers are RECOMMENDED to use globally unique realm values in order to be able to recognize their set of user credentials in a multi-proxy authentication scenario. Implementations MAY use the form "local-realm@host".

The realm value should be considered as an opaque string, which can only be compared for equality with other realms on that server. The server will service the request only if it can validate the user credentials for the protection space of the Request-URI.

EAP packets have a general structure consisting of four basic fields: code, identifier, length and data. The Code field is one octet and it identifies the type of the EAP packet. Packet type is either a request, response, success, or failure. The Identifier field is also one octet and it is used for matching responses with corresponding requests. The Length field is two octets and it indicates in octects the length of the whole EAP packet including code, identifier, length and data fields. The Data field is zero or more octets and its format depends on the content of Code field. The example below demonstrates the general structure of EAP packets.



All these fields (Code, Identifier, Length, and Data) are included in the eap-packet in base64 form. Note that since the packets are self-identifying and self-delimiting it is allowed to include multiple EAP packets within one eap-packet, should some EAP mechanism be able to benefit from this.

Example below demonstrates how a WWW-Authenticate Response Header using EAP authentication would look like:

```
        WWW-Authenticate: eap realm="BollyWorld@example.com",
        eap-p="QWxh4ZGRpb2jpvcGVuNlctZQ=="
```

where "BollyWorld" is the string assigned by the server to identify the protection space of the Request-URI at server "example.com".

A proxy may respond with the same challenge using the Proxy-Authenticate header field. Then it is especially important to

maintain global uniqueness for the realm values, since a request may have credentials for multiple Proxy-Authenticate challenges.

2.2 The Authorization Request Header

In the general HTTP authentication framework, a user agent that wishes to authenticate itself with an origin server or a proxy MAY do so by including an Authorization header or a Proxy-Authorization header field to the request. The authorization field value(s) consists of credentials containing the authentication information of the client for the realm of the resource being requested. The user agent MUST apply the strongest authentication scheme it understands and request credentials from the user based upon the corresponding challenge.

```
credentials     = auth-scheme #auth-param
```

For HTTP EAP Authentication, the framework above is utilized as follows:

```
credentials     = "Eap" eap-response
eap-response    = 1#( realm | eap-param )
eap-param       = "eap-p" "=" eap-packet
eap-packet      = <base64 encoded eap-packet, except
                   not limited to 76 char/line>
```

The value of the realm field must be that supplied in the WWW-Authenticate or Proxy-Authenticate response header for the resource being requested.

Example below demonstrates how the Authorization Request Header using EAP authentication would look like:

```
Authorization: Eap realm="BollyWorld@example.com",
eap-p="QWxhZGRpbjpvcGVuIHNlc2FtZQ=="
```

Rules for handling potential user identifiers, passwords, challenges and so on, are defined in EAP protocol [3].

2.3 Authentication-Info Response Header

The Authentication-Info header is used by the server to communicate information back to the client. This can be either the successful authentication in the response, or the continuation of the EAP mechanism.

```
auth-info       = #auth-param
```

For HTTP EAP authentication the framework above is utilized as follows:

```
Auth-info       = eap-packet
```

eap-packet = <base64 encoded eap-packet, except not limited to 76 char/line>

Example below demonstrates how the Authentication-Info Response Header using EAP authentication would look like:

Authentication-Info: QWxhZGRpbjpvcGVuIHNlc2FtZQ==

The semantics of Proxy-Authentication-Info follow those of Authentication-Info. Proxy-Authentication-Info is used by proxy servers in conjunction with the "407 Proxy Authentication Required" response, and the consequent client authorization request.

3 Security Considerations

Very little about the security of HTTP EAP Authentication can be stated without knowing the chosen EAP authentication scheme. Generally speaking, depending on the chosen EAP authentication scheme, HTTP EAP is subject to the same security threats as HTTP Authentication. However, there are some general aspects, which SHOULD be considered when analyzing the security of HTTP EAP Authentication:

1) Authentication of clients: All EAP mechanisms authenticate the client, using a method dependent on the mechanism.
2) Authentication of servers: Some EAP mechanisms also perform mutual authentication.
3) Using the strongest authentication mechanism available: Servers and clients accepting multiple authentication mechanisms should be aware of the possibility of ’bidding-down’ attacks where a man-in-the-middle modifies the authentication offers until the peers agree on an easily breakable mechanism. In general, we expect HTTP EAP _based servers to require a predefined authentication mechanism from a particular client in any case, which avoids this problem. For instance, the user data base at a server indicates that user A has a particular public key. The server should then insist on using the EAP TLS [4] mechanism to authenticate the user.
4) Confidentiality: Each EAP mechanism offers its specific protection schemes for the exchanged credentials. For instance, the EAP AKA [7] mechanism sends secure cryptographic hashes rather than cleartext passwords like HTTP Basic Authentication does, even if both are based on the concept of a shared secret. As in EAP in general, HTTP EAP does not protect against revealing the identity of the client since the EAP ID RESP packets are not encrypted. Confidentiality and integrity of the HTTP requests themselves beyond the authentication parameters is not within the scope of HTTP EAP, but is discussed below under item 7.
5) Replay protection: Each EAP mechanism offers its specific protection schemes for preventing the replay of the credentials. For instance, the EAP AKA mechanism uses a cryptographically strong sequence number scheme. This is in

contrast to the replay possibilities that exist for the HTTP Basic Authentication, and is similar to the use of nonces in the HTTP Digest Authentication.

6) Integrity protection: Again, each EAP mechanism offers its specific protection schemes against a man-in-the-middle modifying the authentication credentials. Mechanisms based on secure hashes prevent any modifications to the authentication parameters themselves. Again, integrity of the HTTP requests themselves beyond the authentication parameters is a separate issue and is discussed below.
7) Integrity and confidentiality protection of the HTTP request itself is also an important issue. Without such protection, it is possible for a man-in-the-middle to read and modify the actual contents of the request, regardless of any authentication that was performed

Currently, there are no such authentication schemes in HTTP authentication, which would fully protect the integrity of HTTP messages. The HTTP Basic Authentication scheme provides no integrity protection. HTTP Digest Authentication provides only limited (and optional) protection. Most header fields and their values could be modified as part of a man-in-the-middle attack. It should also be noted that HTTP EAP does not inherently provide the integrity protection qualities present in Digest, namely the protection of Request-URI and request-method (and possibly the payload).

Even though HTTP EAP Authentication scheme does not include a protection mechanism, it can be used for setting up one. Chosen EAP authentication scheme may be used to generate session keys, which together with some additional security protocol can provide e.g. integrity protection.

However, such protection should include the protection of original HTTP requests as well. This is not trivial because session protection keys are generated during the authentication, which takes place after submitting the request. In practice, full protection is only possible if the request is repeated at the end of the authentication procedure. This is, however, already the behavior in many typical usage situations. For instance, when authenticating a SIP REGISTER message, the authentication procedure takes a few message rounds, and on each round the REGISTER message is repeated until the session keys are available and the procedure is completed. The last such message can then use integrity protection. Servers that want to avoid man-in-the-middle attacks MUST NOT act on requests until both the authentication procedure has completed and the messages have been received under integrity protection.

4 References

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

2 Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and Stewart, L. "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

3 Blunk, L. and Vollbrecht, J. "PPP Extensible Authentication Protocol (EAP)" RFC 2284, March 1998.

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

5 Aboba, B. "EAP GSS Authentication Protocol" Internet Draft, draft-aboba-pppext-eapgss-08.txt, October 2001.

6 Carlson, J. "PPP EAP SRP-SHA1 Authentication Protocol" Internet Draft, draft-ietf-pppext-eap-srp-03.txt, July 2001.

7 Arkko, J. and Haverinen, H. "EAP AKA Authentication" Internet Draft, draft-arkko-pppext-eap-aka-00.txt, May 2001.

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

2 Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and Stewart, L. “HTTP Authentication: Basic and Digest Access Authentication”, RFC 2617, June 1999.

3 Blunk, L. and Vollbrecht, J. “PPP Extensible Authentication Protocol (EAP)” RFC 2284, March 1998.

4 Aboba, B. and Simon, D. “PPP EAP TLS Authentication Protocol” RFC 2716, October 1999.

5 Aboba, B. “EAP GSS Authentication Protocol” Internet Draft, draft-aboba-pppext-eapgss-03.txt, February 2001.

6 Carlson, J. “PPP EAP SRP-SHA1 Authentication Protocol” Internet Draft, draft-ietf-pppext-eap-srp-01.txt, May 2001.

7 Arkko, J. and Haverinen, H. “EAP AKA Authentication” Internet Draft, draft-arkko-pppext-eap-aka-00.txt, May 2001.

5 Acknowledgements

The authors wish to thank Henry Haverinen and Bernard Aboba for interesting discussions in this problem space.

6 Author's Addresses

Jari Arkko
Ericsson
02420 Jorvas Phone: +358 40 5079256
Finland Email: jari.arkko@ericsson.com

Vesa Torvinen
Ericsson
02420 Jorvas Phone: +358 40 7230822
Finland Email: vesa.torvinen@ericsson.com

Aki Niemi
Nokia Networks
P.O. Box 301
00045 Nokia Group Phone: +358 50 3891644
Finland E-mail: aki.niemi@nokia.com