Network Working Group
Internet-Draft
Expires: November 13, 2002

A. Niemi
Nokia
J. Arkko
V. Torvinen
Ericsson

May 15, 2002

HTTP Digest Authentication Using AKA
draft-ietf-sip-digest-aka-02

Status of this Memo

Copyright Notice

Abstract

The Hypertext Transfer Protocol (HTTP) Authentication Framework includes two authentication schemes: Basic and Digest. Both schemes employ a shared secret based mechanism for access authentication. The Authentication and Key Agreement (AKA) mechanism performs user authentication and session key distribution in Universal Mobile Telecommunications System (UMTS) networks. AKA is a challenge-response based mechanism that uses symmetric cryptography. This memo specifies an AKA based one-time password generation mechanism for HTTP Digest access authentication.

Table of Contents

1. Introduction and Motivation

The Hypertext Transfer Protocol (HTTP) Authentication Framework described in RFC 2617 [2] includes two authentication schemes: Basic and Digest. Both schemes employ a shared secret based mechanism for access authentication. The Basic scheme is inherently insecure in that it transmits user credentials in plain text. The Digest scheme improves security by hiding user credentials with cryptographic hashes, and additionally by providing limited message integrity.

The Authentication and Key Agreement (AKA) [6] mechanism performs authentication and session key distribution in Universal Mobile Telecommunications System (UMTS) networks. AKA is a challenge-response based mechanism that uses symmetric cryptography. AKA is typically run in a UMTS IM Services Identity Module (ISIM), which resides on a smart card like device that also provides tamper proof storage of shared secrets.

This document specifies a mapping of AKA parameters onto HTTP Digest authentication. In essence, this mapping enables the usage of AKA as a one-time password generation mechanism for Digest authentication.

As the Session Initiation Protocol (SIP) [3] Authentication Framework closely follows the HTTP Authentication Framework, Digest AKA is directly applicable to SIP as well as any other embodiment of HTTP Digest.

1.1 Terminology

This chapter explains the terminology used in this document.

AKA
: Authentication and Key Agreement.

AuC
: Authentication Center. The network element in mobile networks that can authorize users either in GSM or in UMTS networks.

AUTN
: Authentication Token. A 128 bit value generated by the AuC, which together with the RAND parameter authenticates the server to the client.

AUTS
: Authentication Token. A 112 bit value generated by the client upon experiencing an SQN synchronization failure.

CK

Cipher Key. An AKA session key for encryption.

IK
Integrity Key. An AKA session key for integrity check.

ISIM
IP Multimedia Services Identity Module.

PIN
Personal Identification Number. Commonly assigned passcodes for use with automatic cash machines, smart cards, etc.

RAND
Random Challenge. Generated by the AuC using the SQN.

RES
Authentication Response. Generated by the ISIM.

SIM
Subscriber Identity Module. GSM counter part for ISIM.

SQN
Sequence Number. Both AuC and ISIM maintain the value of the SQN.

UMTS
Universal Mobile Telecommunications System.

XRES
Expected Authentication Response. In a successful authentication this is equal to RES.

1.2 Conventions

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].

2. AKA Mechanism Overview

This chapter describes the AKA operation in detail:

1. A shared secret K is established beforehand between the ISIM and the Authentication Center (AuC). The secret is stored in the ISIM, which resides on a smart card like, tamper proof device.

2. The AuC of the home network produces an authentication vector AV based on the shared secret K and a sequence number SQN. The

authentication vector contains a random challenge RAND, network authentication token AUTN, expected authentication result XRES, a session key for integrity check IK, and a session key for encryption CK.

3. The authentication vector is downloaded to a server. Optionally, the server can also download a batch of AVs, containing more than one authentication vector.

4. The server creates an authentication request, which contains the random challenge RAND, and the network authenticator token AUTN.

5. The authentication request is delivered to the client.

6. Using the shared secret K and the sequence number SQN, the client verifies the AUTN with the ISIM. If the verification is successful, the network has been authenticated. The client then produces an authentication response RES, using the shared secret K and the random challenge RAND.

7. The authentication response RES is delivered to the server.

8. The server compares the authentication response RES with the expected response XRES. If the two match, the user has been successfully authenticated, and the session keys IK and CK can be used for protecting further communications between the client and the server.

When verifying the AUTN, the client may detect, that the sequence numbers between the client and the server have fallen out of sync. In this case, the client produces a synchronization parameter AUTS, using the shared secret K and the client sequence number SQN. The AUTS parameter is delivered to the network in the authentication response, and the authentication can be tried again based on authentication vectors generated with the synchronized sequence number.

For a specification of the AKA mechanism and the generation of the cryptographic parameters AUTN, RES, IK, CK, and AUTS, see reference 3GPP TS 33.102 [6].

3. Specification of Digest AKA

In general, Digest AKA operation is identical to Digest operation in RFC 2617 [2]. This chapter specifies the parts, in which Digest AKA extends the Digest operation. The notation used in the Augmented BNF definitions for the new and modified syntax elements in this section is as used in SIP [3], and any elements not defined in this section

are as defined in SIP and the documents to which it refers.

3.1 Algorithm Directive

In order to direct the client into using AKA for authentication instead of the standard password system, the RFC 2617 defined algorithm directive is overloaded in Digest AKA:

```
algorithm          =  "algorithm" EQUAL aka-namespace
aka-namespace      =  aka-version "-" algorithm-value
aka-version        =  "AKAv" 1*DIGIT
algorithm-value    =  ( "MD5" / "MD5-sess" / token )
```

algorithm
A string indicating the algorithm used in producing the digest and the checksum. If the directive is not understood, the nonce SHOULD be ignored, and another challenge (if one is present) should be used instead. The default aka-version is "AKAv1". Further AKA versions can be specified, with version numbers assigned by IANA [7]. When the algorithm directive is not present, it is assumed to be "MD5". This indicates, that AKA is not used to produce the Digest password.

Example:

```
algorithm=AKAv1-MD5
```

If the entropy of the used RES value is limited (e.g., only 32 bits), reuse of the same RES value in authenticating subsequent requests and responses is NOT RECOMMENDED. Such a RES value SHOULD only be used as a one-time password and algorithms such as "MD5-sess", which limit the amount of material hashed with a single key by producing a session key for authentication, SHOULD NOT be used.

3.2 Creating a Challenge

In order to deliver the AKA authentication challenge to the client in Digest AKA, the nonce directive defined in RFC 2617 is extended:

```
        nonce              =  "nonce" EQUAL aka-nonce
        aka-nonce          =  LDQUOT aka-nonce-value RDQUOT
        aka-nonce-value    =  <base64 encoding of RAND, AUTN, and
                               server specific data>
```

nonce
A parameter, which is populated with the Base64 [4] encoding of the concatenation of the AKA authentication challenge RAND, the AKA AUTN token, and optionally some server specific data, as in Figure 1.

Example:

```
    nonce="MzQ0a2xrbGtmbGtsZm9wb2tsc2tqaHJzZXNy9uQyMzMzMzQK="
```

Figure 1: Generating the nonce value.

If the server receives a client authentication containing the "auts" parameter defined in Section 3.4 that includes a valid AKA AUTS parameter, the server MUST use it to generate a new challenge to the client. Note that when the AUTS is present, the included "response" parameter is calculated using an empty password (password of ""), instead of a RES.

3.3 Client Authentication

When a client receives a Digest AKA authentication challenge, it

extracts the RAND and AUTN from the "nonce" parameter, and assesses the AUTN token provided by the server. If the client successfully authenticates the server with the AUTN, and determines that the SQN used in generating the challenge is within expected range, the AKA algorithms are run with the RAND challenge and shared secret K.

The resulting AKA RES parameter is treated as "password" when calculating the response directive of RFC 2617.

3.4 Synchronization Failure

For indicating an AKA sequence number synchronization failure, and to re-synchronize the SQN in the AuC using the AUTS token, a new directive is defined for the "digest-response" of the "Authorization" request header defined in RFC 2617:

```
auts            =  "auts" EQUAL auts-param
auts-param      =  LDQUOT auts-value RDQUOT
auts-value      =  <base64 encoding of AUTS>
```

auts
A string carrying a base64 encoded AKA AUTS parameter. This directive is used to re-synchronize the server side SQN. If the directive is present, the client doesn't use any password when calculating its credentials. Instead, the client MUST calculate its credentials using an empty password (password of "").

Example:

```
auts="CjkyMzRfOiwg5CfkJ2UK="
```

Upon receiving the "auts" parameter, the server will check the validity of the parameter value using the shared secret K. A valid AUTS parameter is used to re-synchronize the SQN in the AuC. The synchronized SQN is then used to generate a fresh authentication vector AV, with which the client is then re-challenged.

3.5 Server Authentication

Even though AKA provides inherent mutual authentication with the AKA AUTN token, mutual authentication mechanisms provided by Digest may still be useful in order to provide message integrity.

In Digest AKA, the server uses the AKA XRES parameter as "password"

when calculating the "response-auth" of the "Authentication-Info" header defined in RFC 2617.

4. Example Digest AKA Operation

Figure 2 below describes a message flow describing a Digest AKA process of authenticating a SIP request, namely the SIP REGISTER request.

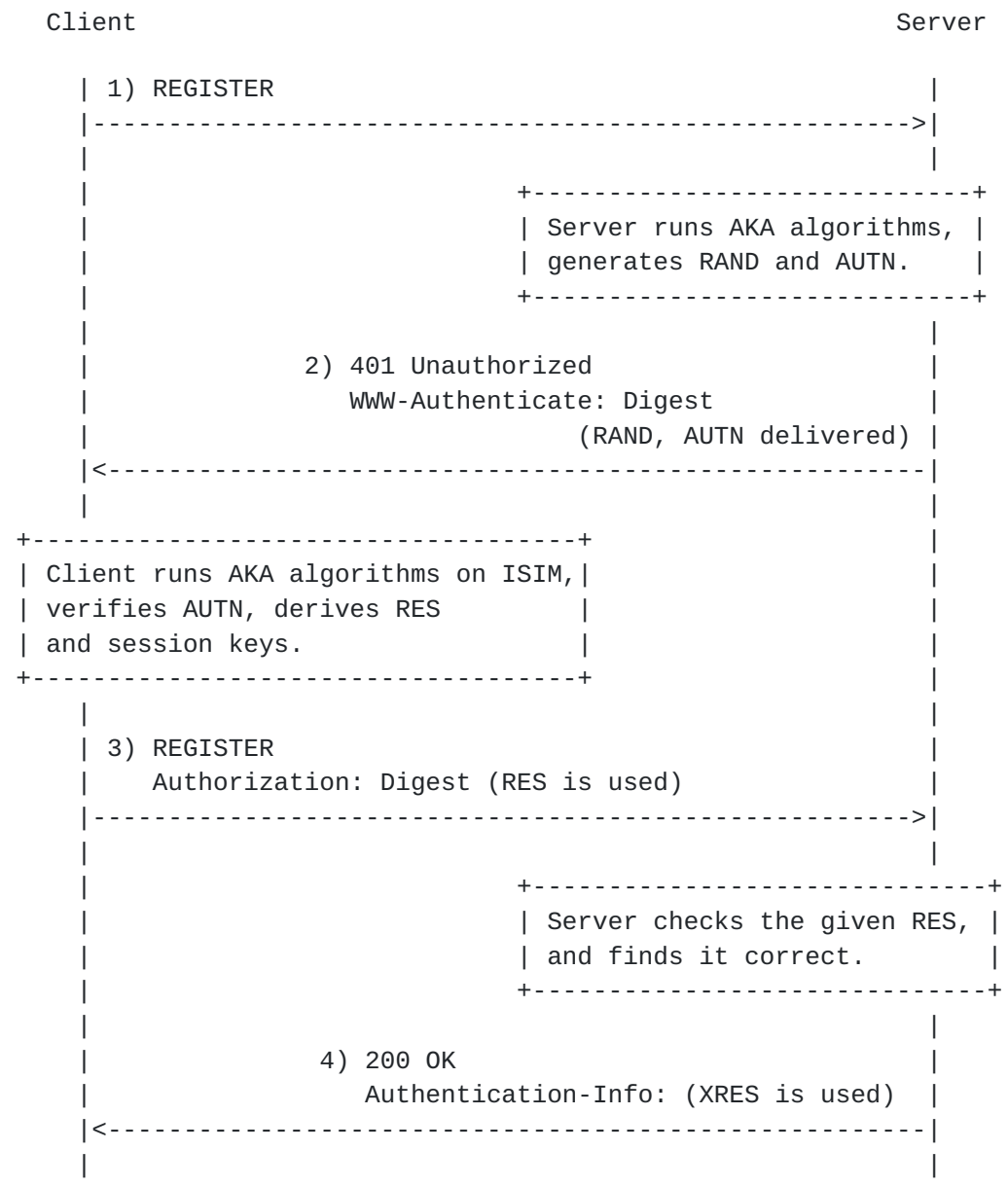


Figure 2: Message flow representing a successful authentication.

1) Initial request

```
REGISTER sip:home.mobile.biz SIP/2.0
```

2) Response containing a challenge

```
SIP/2.0 401 Unauthorized
WWW-Authenticate: Digest
        realm="RoamingUsers@mobile.biz",
        nonce="CjPk9mRqNuT25eRkajM09uTl9nM09uTl9nMz5OX25PZz==",
        qop="auth,auth-int",
        opaque="5ccc069c403ebaf9f0171e9517f40e41",
        algorithm=AKAv1-MD5
```

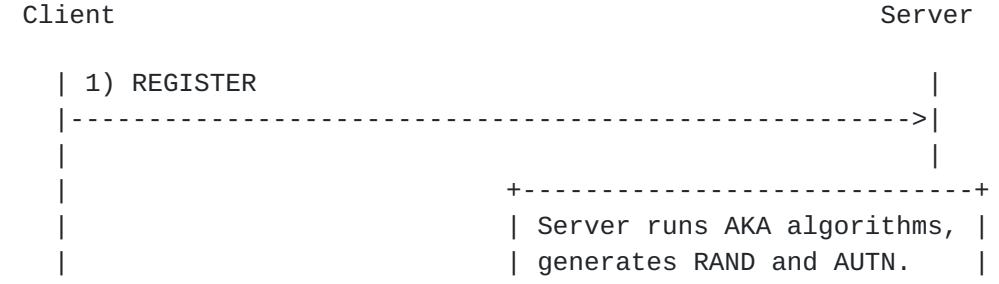
3) Request containing credentials

```
REGISTER sip:home.mobile.biz SIP/2.0
Authorization: Digest
        username="jon.dough@mobile.biz",
        realm="RoamingUsers@mobile.biz",
        nonce="CjPk9mRqNuT25eRkajM09uTl9nM09uTl9nMz5OX25PZz==",
        uri="sip:home.mobile.biz",
        qop=auth-int,
        nc=00000001,
        cnonce="0a4f113b",
        response="6629fae49393a05397450978507c4ef1",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

4) Successful response

```
SIP/2.0 200 OK
Authentication-Info:
        qop=auth-int,
        rspauth="6629fae49393a05397450978507c4ef1",
        cnonce="0a4f113b",
        nc=00000001
```

Figure 3 below describes a message flow describing a Digest AKA authentication process, in which there is a synchronization failure.



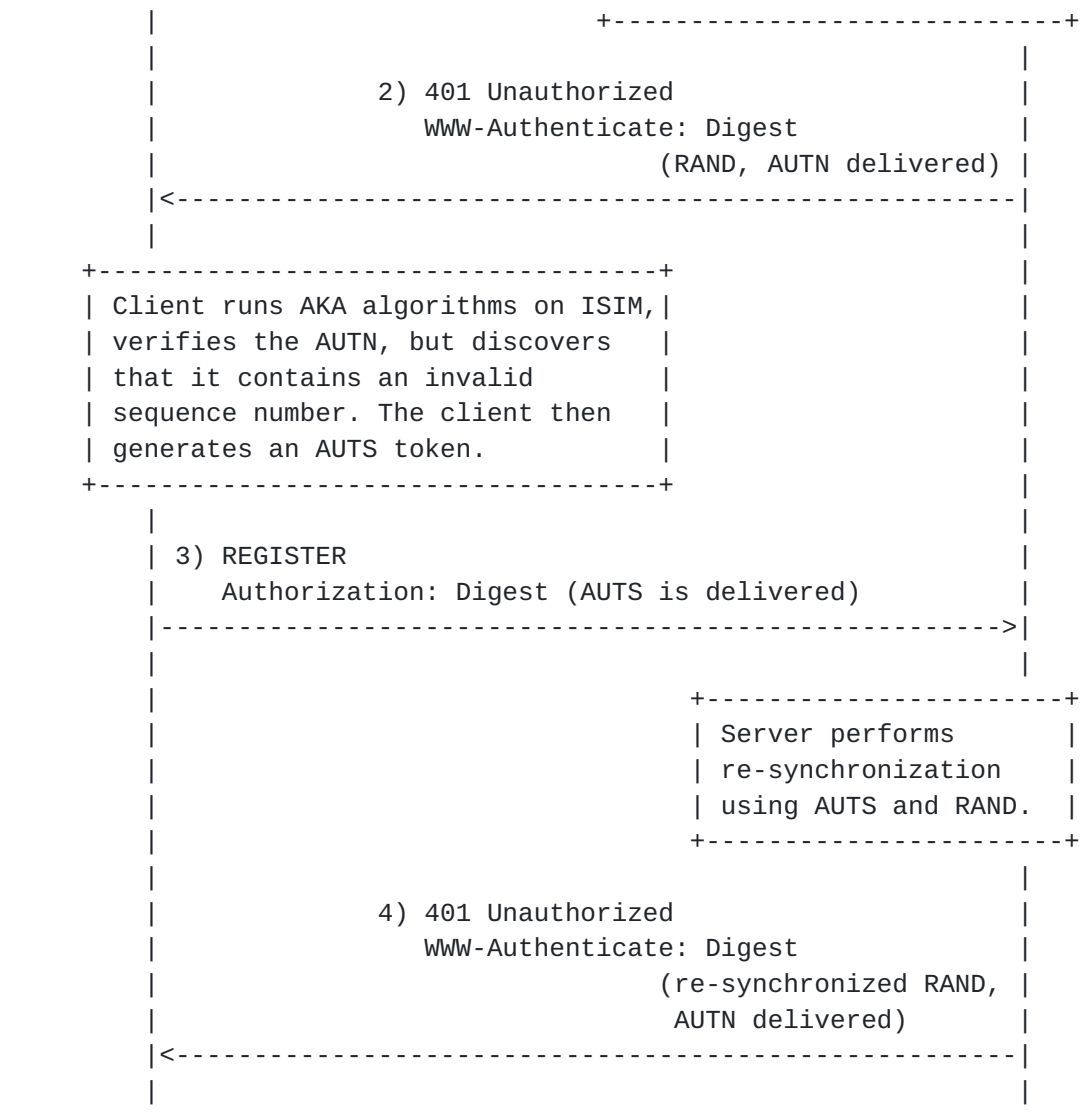


Figure 3: Message flow representing an authentication synchronization failure.

1) Initial request

```
REGISTER sip:home.mobile.biz SIP/2.0
```

2) Response containing a challenge

```
SIP/2.0 401 Unauthorized
WWW-Authenticate: Digest
      realm="RoamingUsers@mobile.biz",
      qop="auth",
      nonce="CjPk9mRqNuT25eRkajM09uTl9nM09uTl9nMz5OX25PZz==",
      opaque="5ccc069c403ebaf9f0171e9517f40e41",
      algorithm=AKAv1-MD5
```

3) Request containing credentials

```
REGISTER sip:home.mobile.biz SIP/2.0
Authorization: Digest
      username="jon.dough@mobile.biz",
      realm="RoamingUsers@mobile.biz",
      nonce="CjPk9mRqNuT25eRkajM09uTl9nM09uTl9nMz5OX25PZz==",
      uri="sip:home.mobile.biz",
      qop=auth,
      nc=00000001,
      cnonce="0a4f113b",
      response="4429ffe49393c02397450934607c4ef1",
      opaque="5ccc069c403ebaf9f0171e9517f40e41",
      auts="5PYxMuX2NOT2NeQ="
```

4) Response containing a new challenge

```
SIP/2.0 401 Unauthorized
WWW-Authenticate: Digest
      realm="RoamingUsers@mobile.biz",
      qop="auth,auth-int",
      nonce="9uQzNPbk9jM05Pbl5Pbl5DIz9uTl9uTl9jM0NTHk9uXk==",
      opaque="dcd98b7102dd2f0e8b11d0f600bfb0c093",
      algorithm=AKAv1-MD5
```

5. Security Considerations

In general, Digest AKA is vulnerable to the same security threats as HTTP authentication [2]. This chapter discusses the relevant exceptions.

5.1 Authentication of Clients using Digest AKA

AKA is typically -- though this isn't a theoretical limitation -- run on an ISIM application that usually resides in a tamper resistant smart card. Interfaces to the ISIM exist, which enable the host device to request authentication to be performed on the card.

However, these interfaces do not allow access to the long-term secret outside the ISIM, and the authentication can only be performed if the device accessing the ISIM has knowledge of a PIN code, shared between the user and the ISIM. Such PIN codes are typically obtained from user input, and are usually required when the device is powered on.

The use of tamper resistant cards with secure interfaces implies that Digest AKA is typically more secure than regular Digest implementations, as neither possession of the host device nor Trojan Horses in the software give access to the long term secret. Where a PIN scheme is used, the user is also authenticated when the device is powered on. However, there may be a difference in the resulting security of Digest AKA, compared to traditional Digest implementations, depending of course on whether those implementations cache/store passwords that are received from the user.

5.2 Limited Use of Nonce Values

The Digest scheme uses server-specified nonce values to seed the generation of the request-digest value. The server is free to construct the nonce in such a way, that it may only be used from a particular client, for a particular resource, for a limited period of time or number of uses, or any other restrictions. Doing so strengthens the protection provided against, for example, replay attacks.

Digest AKA limits the applicability of a nonce value to a particular ISIM. Typically, the ISIM is accessible only to one client device at a time. However, the nonce values are strong and secure even though limited to a particular ISIM. Additionally, this requires that the server is provided with the client identity before an authentication challenge can be generated. If a client identity is not available, an additional round trip is needed to acquire it. Such a case is analogous to an AKA synchronization failure.

A server may allow each nonce value to be used only once by sending a next-nonce directive in the Authentication-Info header field of every response. However, this may cause a synchronization failure, and consequently some additional round trips in AKA, if the same SQN space is also used for other access schemes at the same time.

5.3 Multiple Authentication Schemes and Algorithms

In HTTP authentication, a user agent MUST choose the strongest authentication scheme it understands and request credentials from the user based upon that challenge.

In general, using passwords generated by Digest AKA with other HTTP

authentication schemes is not recommended even though the realm values or protection domains would coincide. In these cases, a password should be requested from the end-user instead. Digest AKA passwords MUST NOT be re-used with such HTTP authentication schemes, which send the password in clear. In particular, AKA passwords MUST NOT be re-used with HTTP Basic.

The same principle must be applied within a scheme if several algorithms are supported. A client receiving an HTTP Digest challenge with several available algorithms MUST choose the strongest algorithm it understands. For example, Digest with "AKAv1-MD5" would be stronger than Digest with "MD5".

5.4 Online Dictionary Attacks

Since user-selected passwords are typically quite simple, it has been proposed that servers should not accept passwords for HTTP Digest, which are in the dictionary [2]. This potential threat does not exist in HTTP Digest AKA because the algorithm will use ISIM originated passwords. However, the end-user must still be careful with PIN codes. Even though HTTP Digest AKA password requests are never displayed to the end-user, she will be authenticated to the ISIM via a PIN code. Commonly known initial PIN codes are typically installed to the ISIM during manufacturing and if the end-users do not change them, there is a danger that an unauthorized user may be able to use the device. Naturally this requires that the unauthorized user has access to the physical device, and that the end-user has not changed the initial PIN code. For this reason, end-users are strongly encouraged to change their PIN codes when they receive an ISIM.

5.5 Session Protection

Digest AKA is able to generate additional session keys for integrity (IK) and confidentiality (CK) protection. Even though this document does not specify the use of these additional keys, they may be used for creating additional security within HTTP authentication or some other security mechanism.

5.6 Replay Protection

AKA allows sequence numbers to be tracked for each authentication, with the SQN parameter. This allows authentications to be replay protected even if the RAND parameter happened to be the same for two authentication requests. More importantly, this offers additional protection for the case where an attacker replays an old authentication request sent by the network. The client will be able to detect that the request is old, and refuse authentication. This

proves liveliness of the authentication request even in the case where a MitM attacker tries to trick the client into providing an authentication response, and then replaces parts of the message with something else. In other words, a client challenged by Digest AKA is not vulnerable for chosen plain text attacks. Finally, frequent sequence number errors would reveal an attack where the tamper-resistant card has been cloned and is being used in multiple devices.

The downside of sequence number tracking is that servers must hold more information for each user than just their long-term secret, namely the current SQN value. However, this information is typically not stored in the SIP nodes, but in dedicated authentication servers instead.

5.7 Improvements to AKA Security

Even though AKA is perceived as a secure mechanism, Digest AKA is able to improve it. More specifically the AKA parameters carried between the client and the server during authentication may be protected along with other parts of the message, by using Digest AKA. This is not possible with plain AKA.

6. IANA Considerations

(This section is not applicable until this document is published as an RFC.)

This document specifies an aka-version namespace in Section 3.1 which requires a central coordinating body. The body responsible for this coordination is the Internet Assigned Numbers Authority (IANA).

The default aka-version defined in this document is "AKAv1". Following the policies outlined in [5], versions above 1 are allocated as Expert Review.

Registrations with the IANA MUST include the version number being registered, including the "AKAv" prefix. For example, a registration for "AKAv2" would potentially be a valid one, whereas a registration for "FOOv2" or "2" would not be valid. Further, the registration MUST include contact information for the party responsible for the registration.

As this document defines the default aka-version, the initial IANA registration for aka-version values will contain an entry for "AKAv1".

6.1 Registration Template

To: ietf-digest-aka@iana.org
Subject: Registration of a new AKA version

Version identifier:

(Must contain a valid aka-version value,
as described in section 3.1.)

Person & email address to contact for further information:

(Must contain contact information for the
person(s) responsible for the registration.)

Normative References

[1] 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 L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[3] Rosenberg, J. and H. Schulzrinne, "SIP: Session Initiation Protocol", draft-ietf-sip-rfc2543bis-09 (work in progress), February 2002.

[4] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

Informative References

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

[6] 3rd Generation Partnership Project, "Security Architecture (Release 4)", TS 33.102, December 2001.

[7] http://www.iana.org, "Assigned Numbers", February 2002.

Authors' Addresses

Aki Niemi
Nokia
P.O. Box 301
NOKIA GROUP, FIN 00045
Finland

Phone: +358 50 389 1644
EMail: aki.niemi@nokia.com

Jari Arkko
Ericsson
Hirsalantie 1
Jorvas, FIN 02420
Finland

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

Vesa Torvinen
Ericsson
Joukahaisenkatu 1
Turku, FIN 20520
Finland

Phone: +358 40 7230822
EMail: vesa.torvinen@ericsson.fi

Appendix A. Acknowledgements

The authors would like to thank Sanjoy Sen, Jonathan Rosenberg, Pete McCann, Tao Haukka, Ilkka Uusitalo, Henry Haverinen, John Loughney, Allison Mankin and Greg Rose.

Full Copyright Statement

Acknowledgement

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