Security Mechanism Agreement for SIP Sessions

1. 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 made obsolete 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 may be found at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories may be found at http://www.ietf.org/shadow.html.

The distribution of this memo is unlimited. It is filed as <draft-ietf-sip-sec-agree-00.txt>, and expires October, 2002. Please send comments to the author or to SIPPING or SIP working group.

2. Abstract

SIP has a number of security mechanisms for hop-by-hop and end-to-end protection. Some of the security mechanisms have been built in to the SIP protocol, such as HTTP authentication or secure attachments. In these mechanisms there are even alternative algorithms and parameters. Currently it isn’t possible to select which security mechanisms to use over a connection. In particular, even if some mechanisms such as OPTIONS were used to make this selection, the selection would be vulnerable against the Bidding-Down attack. This document defines a header for negotiating the security mechanisms within SIP. A SIP entity applying this mechanism must always require some minimum security (i.e. integrity protection) from all communicating parties in order to secure the negotiation, but the negotiation can agree on which specific minimum security is used.
3. Contents

1. Status of this Memo..............................................1
2. Abstract.............................................................1
3. Contents..................................................................2
4. Introduction...........................................................2
5. The Problem.............................................................3
6. Solution...................................................................4
6.1. Requirements..........................................................4
6.2. Different Mechanisms.............................................5
6.3. Overview of Operations..........................................5
6.4. Header descriptions...............................................7
7. Summary of header usage.........................................9
8. Backwards Compatibility........................................10
9. Examples..................................................................10
9.1. Selecting Between New and Old Mechanisms..............10
9.2. Selections Along the Path.......................................11
10. Security Considerations..........................................13
11. IANA Considerations................................................13
12. Modifications..........................................................14
13. Acknowledgments...................................................15

4. Introduction

Traditionally, security protocols have included facilities to agree on the used mechanisms, algorithms, and other security parameters. The reason for this is that experience has shown that e.g. algorithm development uncovers problems in old algorithms and produces new ones. Furthermore, different mechanisms and algorithms are suitable for different situations. Typically, protocols also select other parameters beyond algorithms at the same time.

The purpose of this specification is to define a similar negotiation functionality in SIP [1]. SIP has some security functionality built-in such as HTTP Digest authentication [4], secure attachments such as S/MIME [5], and can also use underlying security protocols such as IPSec/IKE [2] or TLS [3]. Some of the built-in security functionality allows also alternative algorithms and other parameters. While some work within the SIP Working Group has been looking towards reducing the number of recommended security solutions (e.g. recommend just one lower layer security protocol), we can not expect to cut down the number of items in the whole list to one. There will still be multiple security solutions utilized by SIP. Furthermore, it is likely that new methods will appear in the future, to complete the methods that exist today.

Chapter 5 shows that without a secured method to choose between security mechanisms and/or their parameters, SIP is vulnerable to certain attacks. As the HTTP authentication RFC [4] points out, authentication and integrity protection using multiple alternative methods and algorithms is vulnerable to Man-in-the-Middle (MITM) attacks. More seriously, it is hard or sometimes even impossible to know whether a SIP peer entity is truly unable to perform e.g. Digest, TLS, or S/MIME,
or if a MITM attack is in action. In small networks consisting of
workstations and servers these issues are not very relevant, as the
administrators can deploy appropriate software versions and set up
policies for using exactly the right type of security. However, SIP
will soon be deployed to hundreds of millions of small devices with
little or no possibilities for coordinated security policies, let
alone software upgrades, and this makes these issues much worse. This
conclusion is also supported by the requirements from 3GPP [6].

Chapter 6 documents the proposed solution, and chapter 7 gives some
demonstrative examples.

5. The Problem

SIP has alternative security mechanisms such as HTTP authentication /
integrity protection, lower layer security protocols, and S/MIME. It
is likely that their use will continue in the future. SIP security is
developing, and is likely to see also new solutions in the future.

Deployment of large number of SIP-based consumer devices such as 3GPP
terminals requires all network devices to be able to accommodate past,
current and future mechanisms; there is no possibility for instantaê
aneous change since the new solutions are coming gradually in as new
standards and product releases occur. It is sometimes even impossible
to upgrade some of the devices without getting completely new hardê
ware.

So, the basic security problem that such a large SIP-based network
must consider, would be on how do security mechanisms get selected? It
would be desirable to take advantage of new mechanisms as they become
available in products.

Firstly, we need to know somehow what security should be applied, and
preferably find this out without too many additional roundtrips.

Secondly, selection of security mechanisms MUST be secure. Traditionê
ally, all security protocols use a secure form of negotiation. For
instance, after establishing mutual keys through Diffie-Hellman, IKE
sends hashes of the previously sent data -- including the offered
crypto mechanisms. This allows the peers to detect if the initial,
unprotected offers were tampered with.

The security implications of this are subtle, but do have a fundamenê
tal importance in building large networks that change over time. Given
that the hashes are produced also using algorithms agreed in the first
unprotected messages, one could ask what the difference in security
really is. Assuming integrity protection is mandatory and only secure
algorithms are used, we still need to prevent MITM attackers from modê
ifying other parameters, such as whether encryption is provided or
not. Let us first assume two peers capable of using both strong and
weak security. If the initial offers are not protected in any way, any
attacker can easily "downgrade" the offers by removing the strong
options. This would force the two peers to use weak security between
them. But if the offers are protected in some way -- such as by hashing, or repeating them later when the selected security is really on -- the situation is different. It would not be sufficient for the attacker to modify a single message. Instead, the attacker would have to modify both the offer message, as well as the message that contains the hash/repetition. More importantly, the attacker would have to forge the weak security that is present in the second message, and would have to do so in real time between the sent offers and the later messages. Otherwise, the peers would notice that the hash is incorrect. If the attacker is able to break the weak security, the security method and/or the algorithm should not be used.

In conclusion, the security difference is making a trivial attack possible versus demanding the attacker to break algorithms. An example of where this has a serious consequence is when a network is first deployed with integrity protection (such as HTTP Digest [4]), and then later new devices are added that support also encryption (such as S/MIME [1]). In this situation, an insecure negotiation procedure allows attackers to trivially force even new devices to use only integrity protection.

6. Solution

6.1. Requirements

The solution to the SIP security negotiation problem should have the following properties:

(a) It allows the selection of security mechanisms, such as lower layer security protocols or secure attachments. It also allows the selection of individual algorithms and parameters when the security functions are integrated in SIP (such as in the case of HTTP authentication or secure attachments).

(b) It allows both end-to-end and hop-by-hop negotiation.

(c) It is secure, e.g. prevents the bidding down attack.

(d) It is capable of running without additional roundtrips. This is important in the cellular environment, where an additional roundtrip could delay the call set up for 1000-1500 ms.

(e) It does not introduce any additional state to servers and proxies.

Currently, SIP does not have any mechanism which fulfills all the requirements above. The basic SIP features such as OPTIONS and Require, Supported headers are capable of informing peers about various capabilities including security mechanisms. However, the straightforward use of these features cannot guarantee a secured agreement. HTTP Digest algorithm lists [4] are not secure for picking among the digest integrity algorithms, as is described in the RFC itself. More seriously, they have no provisions for allowing encryption to be negotiated. Hence, it would be hard to turn on possible future encryption schemes in a secure manner.
6.2. Different Mechanisms

A self-describing security mechanism is a security mechanism that, when used, contains all necessary information about the method being used as well as all of its parameters such as algorithms.

A non-self-describing security mechanism is a security mechanism that, when used, requires that the use of the method or some of its parameters have been agreed beforehand.

Most security mechanisms used with SIP are self-describing. The use of HTTP digest, as well as the chosen algorithm is visible from the HTTP authentication headers. The use of S/MIME is indicated by the MIME headers, and the CMS structures inside S/MIME describe the used algorithms. TLS is run on a separate port in SIP, and where IPsec/IKE is used, IKE negotiates all the necessary parameters.

The only exception to this list is the use of manually keyed IPsec. IPsec headers do not contain information about the used algorithms. Furthermore, peers have to set up IPsec Security Associations before they can be used to receive traffic. In contrast S/MIME can be received even if no Security Association was in place, because the application can search for a Security Association (or create a new one) after having received a message that contains S/MIME.

In order to make it possible to negotiate both self-describing and non-self-describing security mechanisms, the security agreement scheme must allow both sides to decide on the desired security mechanism before it is actually used. This decision can, and must, take place on both sides before we can be sure that the negotiation has not been tampered by a man-in-the-middle. This tampering will be detected later.

6.3. Overview of Operations

This specification uses the following approach. The clients and servers offer their lists of supported security mechanisms and parameters in the first, unprotected messages. They then proceed to turn on the selected security, and finally repeat some information under this security in order to ensure that the first exchanged lists had not been modified. This procedure is stateless for servers (unless the used security mechanisms require the server to keep some state).

The client and the server lists are both static i.e. they do not and can not change based on the input from the other side. Nodes MAY, however, maintain several static lists, one for each interface, for example.

1. Client --------client list-------> Server
2. Client <--------server list---------- Server
3. Client ------(turn on security)------- Server
4. Client --------server list-------------> Server
5. Client <--------ok or error---------- Server
The steps are explained below:

- Step 1: The client MUST announce a list of supported security mechanisms in their first request. The client SHOULD also add the option-tag ‘sec-agree’ to the Supported header.

- Step 2: The server MUST announce a list of supported security mechanisms in their first response. The server MUST add its list to the response even if there were no common security mechanisms in the client’s and server’s lists, and the list MUST NOT depend on the contents of the client’s list. The list MUST also be added regardless of any potential error codes in the response.

An error has occurred if the client list is not present in the request of Step 1, this request is not OPTIONS, and the server’s policy requires the use of this specification. This error will be reported by returning 421 (Extension Required). In this case the server MUST also include a Require header with an option-tag ‘sec-agree’ in its response.

- Step 3: The peers MUST initiate the security mechanism, if necessary to carry this outside the request in Step 4. For instance, TLS should be turned on at this stage.

- Step 4: The client MUST select and use the first matching security mechanism from the server’s list. The client MUST also repeat the server’s list.

- Step 5: The server MUST check that the server list sent in the secured message in Step 4 corresponds to its static list of supported security mechanisms. The server MUST send a positive answer or proceed to execute the requested operation if and only if the list was not modified. If modification of the list is detected, the server MUST return a 494 (Security Agreement Failed) response or disconnect. The 494 response MUST include server’s unmodified list of supported security mechanisms. The server MUST NOT copy any Security-Mechanism header from the request in Step 4 to the 494 response in Step 5.

The server MAY decide to use a security mechanism between Steps 1 and 2 but it MUST do so before processing request in Step 4. The client MUST decide to use a security mechanism between Steps 2 and 3.

Between Steps 1 and 2, the server MAY set up a non-self-describing security mechanism if necessary. Note that with this type of security mechanisms, the server is necessarily stateful. Between Steps 2 and 4, the client MAY set up a non-self-describing security mechanism.

Note that non-adjacent SIP entities can not use hop-by-hop security mechanisms such as TLS or IPsec. If a client receives a list of hop-by-hop security mechanisms from a server several hops away, it MUST NOT try to use these mechanisms with the first hop proxy. The client MAY try to contact the server directly leaving the proxies in between.
Once the security has been negotiated between two SIP entities, the same SIP entities MAY use the same security when communicating with each other in different SIP roles. For example, if a UAC in an end-user equipment and a UAS in a proxy negotiate some security, they may try to use the same security for terminating requests.

The results of the security mechanism negotiation MAY be informed to the user of an UAC or an UAS. The user MAY decline to accept a particular security mechanism, and abort further SIP communications with the peer.

One SIP request MAY include several independent list. Only one list SHOULD be used between two SIP entities. This allows a negotiation of the first-hop security mechanism while at the same time running e.g. a REGISTER with Digest authentication to a server some hops away.

6.4. Header descriptions

The Security-Mechanism header indicates who wants security towards whom, and what kind of security. The security features are represented using the header syntax described below.

The following ABNF describes the syntax of this header and extends section 25.1 in [1]:

"Security-Mechanism" HCOLON to-uri from-uri mechanism-list

to-uri = "to" EQUAL sip-uri COMMA
from-uri = "from" EQUAL sip-uri COMMA
mechanism-list = 1*(COMMA mechanism [SEMI preference] [SEMI algorithm] [SEMI params])
mechanism = "mech" EQUAL ( "digest" / "tls" / "ipsec-ike" / "ipsec-man" / "smime" / token )
preference = "pref" EQUAL preference-value
algorithm = "alg" EQUAL algorithm-value
params = parameter *[SEMI parameter]
parameter = param-name EQUAL param-value
param-name = token
param-value = token / quoted-string

The meaning of these fields is as follows:

- **to-uri**: Indicates the desired receiver of the information. The value of this field should be a SIP URI. When sent by a client, the value would typically (but not necessarily) contain just the host and port number parts.

- **from-uri**: Indicates the sender of the security agreement information. The value of this is also a SIP URI. When sent by a client, the value would typically
The security mechanism supported by the SIP entity identified in the from-uri.

This specification defines six values:

- "tls" for TLS [3],
- "digest" for HTTP Digest [4],
- "ipsec-ike" for IPsec with IKE [2],
- "ipsec-man" for manually keyed IPsec without IKE,
- "smime" for S/MIME [5], and
- token for extensions

To use TLS, the client MUST contact the server side on port 5061. If this connection attempt fails, the security agreement procedure MUST be considered to have failed, and MUST be terminated.

To use HTTP Digest, it alone does not fulfill the minimum security requirements of this specification. In order to use HTTP Digest securely, some variant of MIME tunneling SHOULD be used to force the Security-Mechanism header to be integrity protected in the MIME body. Also, if the server decides that the first matching mechanism is HTTP digest in Step 2 of Section 6.3, the server SHOULD include a HTTP authentication challenge in its response. However, HTTP Digest need not to be negotiated if some underlying security is used (e.g. TLS or IPsec). The proxy/server can always challenge the client after some security mechanism is already in place.

To use either "ipsec-ike" or "ipsec-man", the client and the server MUST request their IPsec implementations to protect all further communications between the same IP addresses and ports which were used in the first request from the client and first response from the server. If the IKE connection attempt fails, the agreement procedure MUST be considered to have failed, and MUST be terminated. Clients and servers SHOULD terminate the IPsec protection when it is no longer used.

Note also that "ipsec-man" will only work if the communicating SIP entities know which keys and other parameters to use. It is outside the scope of this specification to describe how this information can be made known to the peers.

To use S/MIME, the client MUST construct its request in Step 4 (see 6.3) using S/MIME. If the server sees that S/MIME is the selected mechanism in Step 2, it MAY send its own certificate within an S/MIME body in the response of Step 2.
preference  A positive integer identifying the preferred order of the mechanisms. Servers MUST use preference numbers in their lists to identify the preferred order of the security mechanisms. Clients MUST NOT use preference numbers in their lists.

algorithm  An optional algorithm field for those security mechanisms which are not self-describing or which are vulnerable for bidding-down attacks (e.g. HTTP Digest). In the case of HTTP Digest, the same rules apply as defined in [4] for the "algorithm" field in HTTP Digest.

params  An optional field that allows future extensions. Any unrecognized directive MUST be ignored.

Multiple instances of the same header field can appear in SIP messages. Typically, the client inserts its own Security-Mechanism header when it sends a request, and the server/proxy adds its own to the response. The parameters are in all cases set in an appropriate manner to indicate in the "to-uri" parameter the party who inserted the header. Or rather -- since the client is copying some of the server’s responses -- whose security capabilities the header applies to.

7. Summary of header usage

The Security-Mechanism header may be used to negotiate the security mechanisms between various SIP entities including UAC, UAS, proxy, and registrar. Information about the use of Security-Mechanism header in relation to SIP methods and proxy processing is summarized in Table 1.

<table>
<thead>
<tr>
<th>Header field</th>
<th>where</th>
<th>proxy</th>
<th>ACK</th>
<th>BYE</th>
<th>CAN</th>
<th>INV</th>
<th>OPT</th>
<th>REG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security-Mechanism</td>
<td>R</td>
<td>ar</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>Security-Mechanism</td>
<td>2xx</td>
<td>a</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>Security-Mechanism</td>
<td>401, 407, 421, 494</td>
<td>a</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
</tbody>
</table>

Table 1: Summary of header usage.

The "where" column describes the request and response types in which the header field may be used. The header may not appear in other types of SIP messages. Values in the where column are:

- R: Header field may appear in requests.

- 2xx, 401, etc.: A numerical value or range indicates response codes with which the header field can be used.

The "proxy" column describes the operations a proxy may perform on a header field:
- a: A proxy can add or concatenate the header field if not present.

- r: A proxy must be able to read the header field, and thus this header field cannot be encrypted.

The next six columns relate to the presence of a header field in a method:

- c: Conditional; requirements on the header field depend on the context of the message.

8. Backwards Compatibility

SIP entities which support this specification but whose policy does not require its use, SHOULD only start using it if so required by the peer. Such SIP entities can thus communicate with other SIP entities even if they do not support this specification.

SIP entities that support this specification and have a policy which requires its use MUST insert the Supported and Require headers as described in this specification. This makes communications possible only with nodes that support this specification. The OPTIONS method can still be used with any node.

Note that the use of this specification together with the Proxy-Require header demands the support of this specification from all SIP entities along the path.

9. Examples

9.1. Selecting Between New and Old Mechanisms

In this example we demonstrate the use of the framework for securing a hop using some security mechanism, without knowing beforehand which methods the server supports. We assume that the client is not willing to reveal any information on what it intends to do, so it uses OPTIONS in the first message that is sent in the clear. The example starts by a client sending a message to the server, indicating that it is of the new variant that supports TLS in Step 1. In Step 2, the server responds that with its own list of security mechanisms -- S/MIME or TLS in this case -- and the peers start only common security service i.e. TLS at Step 3. In Step 4, the client resends the server’s Security-Mechanism header, which the server verifies, and responds with 200 OK.

1. Client -> Server:

   OPTIONS server SIP/2.0
   Security-Mechanism: to=sip:server.example.com,
                        from=sip:client.example.com,
                        mech=tls

2. Server -> Client:

   200 OK
Security-Mechanism: to=sip:client.example.com, from=sip:server.example.com, mech=smime;pref=1, mech=tls;pref=2

3. Security handshake at a lower layer i.e. TLS

4. Client -> Server:

INVITE server SIP/2.0
Security-Mechanism: to=sip:client.example.com, from=sip:server.example.com, mech=smime;pref=1, mech=tls;pref=2

5. Server -> Client:

200 OK

In the example we have omitted the returned values of Security-Mechanism in replies for clarity. Typically in SIP the servers do not remove header fields as they answer, they only add new headers.

If this example was run without Security-Mechanism in Step 2, the client would not know what kind of security the other one supports, and would be forced to error-prone trials.

More seriously, if the Security-Mechanism was omitted in Step 4, the whole process would be prone for MITM attacks. An attacker could spoof "ICMP Port Unreachable" message on the trials, or remove the stronger security option from the header in Step 1, therefore substantially reducing the security.

9.2. Selections Along the Path

This example attempts to show how selections can be made between a client and the first-hop proxy while the actual SIP messages are still destined to a server further on in the network. This example demonstrates how we can securely agree on the security mechanism between the client and its first hop proxy, without adding roundtrips.

In this example, the client sends a REGISTER request to its registrar. At the same time, the client negotiates the security with a different first hop proxy. There are three alternative security solutions: a) TLS, b) IPsec/IKE, and c) HTTP Digest.

The example starts by a client coming to a new network and learning the address of the local proxy. The proxy is of an upgraded version, so it supports all security mechanisms. The client supports alternatives b) and c). The client also knows the address of the registrar. We assume that some trust has already been established between the client and the home, and between the client and the proxy. Perhaps this trust is in the form of the nodes belonging under the same PKI, or having distributed shared secrets beforehand.
In Step 1 the client contacts the proxy using a REGISTER message. We omit the details of the communications with the home server in this discussion, but the proxy forwards the messages onwards in Step 2. In Step 3, the registrar responds with an end-to-end HTTP Digest authentication challenge. Using the same response, the proxy adds an indication that it supports TLS with HTTP Digest, IPsec/IKE, and plain HTTP Digest. In Step 4, the client selects the first method it supports (IPsec/IKE in this case), the protection is turned on. In Step 5, the client sends the next round of REGISTER messages to the registrar. This message includes the repetition of the original security capabilities of the proxy. The proxy verifies this list, and forwards the request to the registrar. In Step 7, the registrar responds with a 200 OK.

1. Client -> Proxy:

   REGISTER server SIP/2.0
   Security-Mechanism: to=sip:proxy.example.com,
   from=sip:client.example.com,
   mech=ipsec-ike, mech=digest;alg=MD5

2. Proxy communicates with the Server.

3. Proxy -> Client:

   401 Authentication Required
   (HTTP Digest challenge from the registrar to the client)
   Security-Mechanism: to=sip:client.example.com,
   from=sip:proxy.example.com,
   mech=tls;pref=1, mech=ipsec-ike;pref=2,
   mech=digest;pref=3;alg=MD5

4. Security handshake at a lower layer i.e. IPsec/IKE

5. Client -> Proxy:

   REGISTER server SIP/2.0
   Security-Mechanism: to=sip:client.example.com,
   from=sip:proxy.example.com,
   mech=tls;pref=1, mech=ipsec-ike;pref=2,
   mech=digest;pref=3;alg=MD5

6. Proxy communicates with the Server.

7. Proxy -> Client:

   200 OK

As in the previous example, if this was run without Security-Mechanism in Step 3, the client would not know what kind of algorithms the server supports. In this example we demonstrate also the need for the client to send its own mechanism list in Step 1. If this wasn’t known
to the proxy when it responds in Step 3, it could not have provided a suitable HTTP Digest challenge because at that point the proxy would not have known if the client supports that.

As in the previous example, removing the repetition of the Security-Mechanism header in Step 5 would open the system to MITM attacks.

10. Security Considerations

This specification is about making it possible to select between various SIP security mechanisms in a secure manner. In particular, the method presented here allow current networks using e.g. Digest later securely upgrade to e.g. S/MIME without requiring a simultaneous modification in all equipment. The method presented in this specification is secure only if the weakest proposed mechanism offers at least integrity protection.

Attackers could try to modify the server’s list of security mechanisms in the first response. This would be revealed to the server when the client returns the received list using the security.

Attackers could also try to modify the repeated list in the second request from the client. However, if the selected security mechanism uses encryption this may not be possible, and if it uses integrity protection any modifications will be detected by the server.

Finally, attackers could try to modify the client’s list of security mechanisms in the first message. The client selects the security mechanism based on its own knowledge of its own capabilities and the server’s list, hence the client’s choice would be unaffected by any such modification. However, the server’s choice could still be affected as described below:

- If the modification affected the server’s choice, the server and client would end up choosing different security mechanisms in Step 3 or 4.) Since they would be unable to communicate to each other, this would be detected as a potential attack. The client would either retry or give up in this situation.

- If the modification did not affect the server’s choice, there’s no effect.

All clients that implement this specification MUST select HTTP Digest, S/MIME, TLS, IPsec, or any stronger method for the protection of the second request. If HTTP Digest is used alone, the security agreement headers MUST be protected. This can be done with HTTP Digest if combined with MIME/SIP tunneling, for example.

11. IANA Considerations

This specification defines ‘sec-agree’ option tag which should be registered in IANA. The option-tag ’sec-agree’ can be used in header related to the SIP compatibility mechanisms for extensions (e.g. Require and Supported).
This specification also defines a new error code, 494 (Security Agreement Failed), which should be registered in IANA.

12. Modifications

The draft-sip-sec-agree-00.txt version of this specification introduced the following modifications:

- Many editorial changes, restructuring and clarifications.
- Motivation section has been shortened since this is now a WG item.
- Clarified that the solution requires always some base level of security (i.e. integrity) in order to work. Even ‘the weak security’ must not be broken.
- Text related to alternative solutions shortened and moved to a new place.
- New rules for possible error and special cases has been added, e.g. for the case in which an non-adjacent SIP entities try to negotiate hop-by-hop security mechanisms.
- Syntax of the header redesigned. Wanted to get rid of the semantics related to the relative position of a header component in the header (e.g. first parameters defines the ‘from-uri’, second the ‘to-uri’, and third the first supported security mechanism). The option tags are now used to identify the Security Agreement extension, not the individual security mechanisms.
- The semantics of the header slightly changed: the AND operator between the individual mechanisms is removed because it is really needed with HTTP Digest only. And even in this case, the negotiation is not needed beforehand if some underlying security is used.
- Options for HTTP Digest algorithms and manually keyed IPsec added.
- Explicit rules were added to all mechanisms on how they should be used, such as TLS to be run on port 5061 etc.
- References to Enhanced HTTP Digest removed.
- Example related to 3GPP generalized.

The draft-arkko-sip-sec-agree-01.txt version of this specification introduced the following modifications:

- Reversed approach to make servers stateless
- Removed discussion of the use of this for Digest algorithm selection, since Enhanced Digest already has bidding-down protection
- Renamed org.iana.sip.digest to org.iana.sip.edigest and removed the parameters, as we can rely on Enhanced Digest to perform the algorithm
selection.
- Removed agreements for full paths.
- Simplified syntax

13. Acknowledgments

The authors wish to thank Rolf Blom, James Undery, Jonathan Rosenberg, Hugh Shieh, Gunther Horn, Krister Boman, David Castellanos-Zamora, Aki Niemi, Miguel Garcia, Valtteri Niemi, Martin Euchner, Eric Rescorla, and members of the 3GPP SA3 group for interesting discussions in this problem space.

14. Normative References


15. Non-Normative References


16. Author's Address

Jari Arkko, Vesa Torvinen
Ericsson
02420 Jorvas
Finland
EMail: Jari.Arkko@ericsson.com, Vesa.Torvinen@ericsson.fi

Tao Haukka
Nokia
Finland
EMail: Tao.Haukka@nokia.com
Sanjoy Sen
Nortel Networks
2735-B Glenville Drive
Richardson, TX 75082, USA
EMail: sanjoy@nortelnetworks.com

Lee Valerius
Nortel Networks
2201 Lakeside Blvd
Richards, TX 75082, USA
EMail: valerius@nortelnetworks.com