**3GPP TSG SA WG3 Security** **S3-020516**

**8th – 11th October, 2002**

**Munich, Germany**

**Agenda Item:** 7.1

**Source:** Ericsson & Nokia

**Title:** IETF status report: SIP security agreement

**Document for:** Discussion/Decision

# 1 Scope and objectives

Ericsson, Nokia and Nortel Networks have been working on draft "Security Mechanism Agreement for SIP Sessions" [Sec-Agree] in IETF. A new, unofficial version of the draft has been submitted to Internet Engineering Steering Group (IESG) security review since the previous SA3 meeting (Helsinki, July 2002). This version of the draft is attached to this document. Note that this version is not publicly available in IETF.

Since the version –04, the major changes to the draft has been:

- The scope has been limited to the first-hop negotiation
- The relationship of the draft to SIPS & DNS processes has been clarified
- S/MIME has been removed (because it is for end-to-end)
- A new procedure and syntax for HTTP Digest has been introduced
- The motivational part of the draft has been rewritten

IESG is still doing the security review on the draft.

Since there is still some uncertainty whether the draft will pass the IESG review, SA3 is recommended to prepare a backup plan. There are several alternatives:

1) Include the draft to the annex of 33.203. This alternative has a drawback of breaking IETF rules for SIP extensions. It also breaks interoperability because the same error code, option tag and header field name may be used by IETF for other purposes.

2) Carry the Security Association parameters in some existing extensible headers, reuse existing error codes, and do not use option tags. This alternative is better in terms of interoperability.

3) Some other variant of 1) and/or 2).

4) Redesign the IMS security solution. In practice, this is not possible in R5 timeframe.

# 2 Proposal

SA3 should decide about the deadline when [Sec-Agree] is not anymore considered as IMS R5 solution if not accepted in IETF. It is proposed that if the document is not approved in IETF during October, SA3 should not consider [Sec-Agree] as IMS R5 solution. Meanwhile, it is proposed that SA3 should discuss on the backup alternatives in this meeting. Alternative solutions should be developed after the meeting as a backup plan. Appropriate solution should be as independent of IETF as possible to make sure there is at least one solution available by SA3#26 in November.

# References

[Sec-Agree] Arkko et al, “Security Mechanism Agreement for SIP Sessions”, IETF, Work in progress, June 2002, draft-ietf-sip-sec-agree-04.txt.

SIP Working Group
INTERNET-DRAFT
<draft-ietf-sip-sec-agree-05.txt>
September 2002
Expires: March 2002

Jari Arkko
Vesa Torvinen
Gonzalo Camarillo
Ericsson
Tao Haukka
Nokia
Sanjoy Sen
Nortel Networks

# Security Mechanism Agreement for the Session Initiation Protocol

## Status of this memo



## Abstract

This document defines new functionality for negotiating the security mechanisms used between a Session Initiation Protocol (SIP) user agent and its next-hop SIP entity. This new functionality supplements the existing methods of choosing security mechanisms between SIP entities.

## TABLE OF CONTENTS

## 1. Introduction

Traditionally, security protocols have included facilities to agree on the used mechanisms, algorithms, and other security parameters. This is to add flexibility, since different mechanisms are usually suitable to different scenarios. Also, the evolution of security mechanisms often introduces new algorithms, or uncovers problems in existing ones, making negotiation of mechanisms a necessity.

The purpose of this specification is to define negotiation functionality for the Session Initiation Protocol (SIP) [1]. This negotiation is intended to work only between a UA and its first-hop SIP entity.

### 1.1. Motivations

Without a secured method to choose between security mechanisms and/or their parameters, SIP is vulnerable to certain attacks. Authentication and integrity protection using multiple alternative methods and algorithms is vulnerable to Man-in-the-Middle (MitM) attacks [see e.g. 4].

It is also hard or sometimes even impossible to know whether a specific security mechanism is truly unavailable to a SIP peer entity, or if in fact a MitM attack is in action.

In certain small networks these issues are not very relevant, as the administrators of such networks can deploy appropriate software versions and set up policies for using exactly the right type of security. However, SIP is also expected to be deployed to hundreds of millions of small devices with little or no possibilities for coordinated security policies, let alone software upgrades, which necessitates the need for the negotiation functionality to be available from the very beginning of deployment [see e.g. 10].

### 1.2. Design Goals

A. The entities involved in the security agreement process need to find out exactly which security mechanisms to apply, preferably without excessive additional roundtrips.

B. The selection of security mechanisms itself needs to be secure. Traditionally, 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 [9]. This allows the peers to detect if the initial, unprotected offers were tampered with.

C. The entities involved in the security agreement process need to be able to indicate success or failure of the security agreement process.

D. The security agreement process should not introduce any additional state to be maintained by the involved entities.

## 2. Solution

### 2.1 Overview of Operations

The message flow below illustrates how the mechanism defined in this document works:

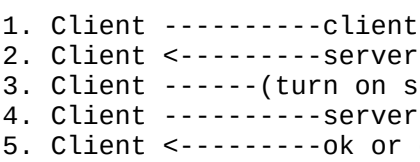


Figure 1: Security agreement message flow

Step 1: Clients wishing to use this specification can send a list of their supported security mechanisms along the first request to the server.

Step 2: Servers wishing to use this specification can challenge the client to perform the security agreement procedure. The security mechanisms and parameters supported by the server are sent along in this challenge.

Step 3: The client then proceeds to select the highest-preference security mechanism they have in common and to turn on the selected security.

Step 4: The client contacts the server again, now using the selected security mechanism. The server’s list of supported security mechanisms is returned as a response to the challenge.

Step 5: The server verifies its own list of security mechanisms in order to ensure that the original list 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 cannot change based on the input from the other side). Nodes may, however, maintain several static lists, one for each interface, for example.

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. The client would set up the non-self-describing security mechanism between Steps 2 and 4.

## 2.2 Syntax

We define three new SIP header fields, namely Security-Client, Security-Server and Security-Verify. The notation used in the Augmented BNF definitions for the syntax elements in this section is as used in SIP [1], and any elements not defined in this section are as defined in SIP and the documents to which it refers:

```
   security-client  = "Security-Client" HCOLON
                      sec-mechanism *(COMMA sec-mechanism)
   security-server  = "Security-Server" HCOLON
                      sec-mechanism *(COMMA sec-mechanism)
   security-verify  = "Security-Verify" HCOLON
                      sec-mechanism *(COMMA sec-mechanism)
   sec-mechanism    = mechanism-name *(SEMI mech-parameters)
   mechanism-name   = ( "digest" / "tls" / "ipsec-ike" /
                       "ipsec-man" / token )
   mech-parameters  = ( preference / digest-algorithm /
                        digest-qop / digest-verify / extension )
   preference       = "q" EQUAL qvalue
   qvalue           = ( "0" [ "." 0*3DIGIT ] )
                       / ( "1" [ "." 0*3("0") ] )
   digest-algorithm = "d-alg" EQUAL token
   digest-qop       = "d-qop" EQUAL token
   digest-verify    = LDQUOT 32LHEX RDQUOT
   extension        = generic-param
```

Note that qvalue is already defined in the SIP BNF [1]. We have copied its definitions here for completeness.

The parameters described by the BNF above have the following semantics:

Mechanism-name
It identifies the security mechanism supported by the client, when it appears in a Security-Client header fields, or by the server, when it appears in a Security-Server or in a Security-Verify header field. This specification defines four 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.

Preference

The "q" value indicates a relative preference for the particular mechanism. The higher the value the more preferred the mechanism is. All the security mechanisms MUST have different "q" values. It is an error to provide two mechanisms with the same "q" value.

Digest-algorithm
This optional parameter is defined here only for HTTP Digest [4] in order to prevent the bidding-down attack for the HTTP Digest algorithm parameter. The content of the field may have same values as defined in RFC 2617 [4] for the "algorithm" field.

Digest-qop
This optional parameter is defined here only for HTTP Digest [4] in order to prevent the bidding-down attack for the HTTP Digest qop parameter. The content of the field may have same values as defined in RFC 2617 [4] for the "qop" field.

Digest-verify
This optional parameter is defined here only for HTTP Digest [4] in order to prevent the bidding-down attack for the SIP security mechanism agreement (this document). The content of the field is counted exactly the same way as "request-digest" in [4] except that the Security-Server header field is included in the A2 parameter. If the "qop" directive's value is "auth" or is unspecified, then A2 is:

A2 = Method ":" digest-uri-value ":" security-server

If the "qop" value is "auth-int", then A2 is:

A2 = Method ":" digest-uri-value ":" H(entity-body) ":" security-server

All linear white spaces in the Security-Server header field MUST be replaced by a single SP before calculating or interpreting the digest-verify parameter. Method, digest-uri-value, entity-body, and any other HTTP Digest parameter are as specified in [4].

Note that this specification does not introduce any extension or change to HTTP Digest [4]. This specification only re-uses the existing HTTP Digest mechanisms to protect the negotiation of security mechanisms between SIP entities.

## 2.3 Protocol Operation

This section deals with the protocol details involved in the negotiation between a SIP UA and its next-hop SIP entity. Throughout the text the next-hop SIP entity is referred to as the first-hop proxy or outbound proxy. However, the reader should bear in mind that a user agent server can also be the next-hop for a a user agent client.

### 2.3.1 Client Initiated

If a client ends up using TLS to contact the server because it has followed the rules specified in [6], the client MUST NOT use the security agreement procedure of this specification. If a client ends up using non-TLS connections because of the rules in [6], the client MAY use the security agreement of this specification to detect DNS spoofing, or to negotiate some other security than TLS.

A client wishing to use the security agreement of this specification MUST add a Security-Client header field to a request addressed to its first-hop proxy (i.e., the destination of the request is the first-hop proxy). This header field contains a list of all the security mechanisms that the client supports. The client SHOULD NOT add preference parameters to this list. The client MUST add both a Require and Proxy-Require header field with the value "sec-agree" to its request.

The contents of the Security-Client header field may be used by the server to include any necessary information in its response.

A server receiving an unprotected request that contains a Require or Proxy-Require header field with the value "sec-agree" MUST respond to the client with a 494 (Security Agreement Required) response. The server MUST add a Security-Server header field to this response listing the security mechanisms that the server supports. The server MUST add its list to the response even if there are no common security mechanisms in the client's and server's lists. The server's list MUST NOT depend on the contents of the client's list.

The server MUST compare the list received in the Security-Client header field with the list to be sent in the Security-Server header field. When the client receives this response, it will choose the common security mechanism with the highest "q" value. Therefore, the server MUST add the necessary information so that the client can initiate that mechanism (e.g., a Proxy-Authenticate header field for HTTP Digest).

When the client receives a response with a Security-Server header field, it MUST choose the security mechanism in the server's list with the highest "q" value among all the mechanisms that are known to the client. Then, it MUST initiate that particular security mechanism as described in Section 3.5. This initiation may be carried out without involving any SIP message exchange (e.g., establishing a TLS connection).

If an attacker modified the Security-Client header field in the request, the server may not include in its response the information needed to establish the common security mechanism with the highest preference value (e.g., the Proxy-Authenticate header field is missing). A client detecting such a lack of information in the response MUST consider the current security agreement process aborted, and MAY try to start it again by sending a new request with a Security-Client header field as described above.

All the subsequent SIP requests sent by the client to that server SHOULD make use of the security mechanism initiated in the previous step. These requests MUST contain a Security-Verify header field that

mirrors the server's list received previously in the Security-Server header field. These requests MUST also have both a Require and Proxy-Require header fields with the value "sec-agree".

The server MUST check that the security mechanisms listed in the Security-Verify header field of incoming requests correspond to its static list of supported security mechanisms.

> Note that, following the standard SIP header field comparison rules defined in [1], both lists have to contain the same security mechanisms in the same order to be considered equivalent. In addition, for each particular security mechanism, its parameters in both lists need to have the same values.

The server can proceed processing a particular request if, and only if, the list was not modified. If modification of the list is detected, the server MUST respond to the client with a 494 (Security Agreement Required) response. This response MUST include the server's unmodified list of supported security mechanisms. If the list was not modified, and the server is a proxy, it MUST remove the "sec-agree" value from both the Require and Proxy-Require header fields, and then remove the header fields if no values remain.

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 and its outbound proxy negotiate some security, they may try to use the same security for incoming requests (i.e., the UA will be acting as a UAS).

The user of a UA SHOULD be informed about the results of the security mechanism agreement. The user MAY decline to accept a particular security mechanism, and abort further SIP communications with the peer.

### 2.3.2 Server Initiated

A server decides to use the security agreement described in this document based on local policy. If a server receives a request from the network interface that is configured to use this mechanism, it must check that the request has only one Via header field. If there are several Via header fields, the server is not the first-hop SIP entity, and it MUST NOT use this mechanism. For such a request, the server must return a 502 (Bad Gateway) response.

A server that decides to use this agreement mechanism MUST challenge unprotected requests with one Via header field regardless of the presence or the absence of any Require, Proxy-Require or Supported header fields in incoming requests.

A server that by policy requires the use of this specification and receives a request that does not have the sec-agree option tag in a Require, Proxy-Require or Supported header field MUST return a 421 (Extension Required) response. If the request had the sec-agree option tag in a Supported header field, it MUST return a 494 (Security Agreement Required) response. In both situation the server

MUST also include in the response a Security-Server header field listing its capabilities and a Require header field with an option-tag "sec-agree" in it. The server MUST also add necessary information so that the client can initiate the preferred security mechanism (e.g., a Proxy-Authenticate header field for HTTP Digest).

Clients that support the extension defined in this document MAY add a Supported header field with a value of "sec-agree".

## 2.4. Security Mechanism Initiation

Once the client chooses a security mechanism from the list received in the Security-Server header field from the server, it initiates that mechanism. Different mechanisms require different initiation procedures.

If "tls" is chosen, the client uses the procedures of Section 8.1.2 of [1] to determine the URI to be used as an input to the DNS procedures of [6]. However, if the URI is a sip URI, it MUST treat the scheme as if it were sips, not sip. If the URI scheme is not sip, the request MUST be sent using TLS.

If "digest" is chosen, the 494 (Security Agreement Required) response will contain an HTTP Digest authentication challenge. The client MUST use the algorithm and qop parameters in the Security-Server header field to replace the same parameters in the HTTP Digest challenge. The client MUST also use the digest-verify parameter to protect the Security-Server header field as specified in 2.2.

To use "ipsec-ike", the client attempts to establish an IKE connection to the host part of the Request-URI in the first request to the server. If the IKE connection attempt fails, the agreement procedure MUST be considered to have failed, and MUST be terminated.

Note 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. All rules for minimum implementations, such as mandatory-to-implement algorithms, apply as defined in [2, 7, and 8].

In both IPsec-based mechanisms, it is expected that appropriate policy entries for protecting SIP have been configured or will be created before attempting to use the security agreement procedure, and that SIP communications use port numbers and addresses according to these policy entries. It is outside the scope of this specification to describe how this information can be made known to the peers, but it would typically be configured at the same time as the IKE credentials or manual SAs have been entered.

## 2.5. Duration of Security Associations

Once a security mechanism has been negotiated, both the server and the client need to know until when it can be used. All the mechanisms described in this document have a different way of signaling the end

of a security association. When TLS is used, the termination of the connection indicates that a new negotiation is needed. IKE negotiates the duration of a security association. If the credentials provided by a client using digest are no longer valid, the server will re-challenge the client. It is assumed that when IPsec-man is used, the same out-of-band mechanism used to distribute keys is used to define the duration of the security association.

## 2.6. Summary of Header Field Use

The header fields defined in this document may be used to negotiate the security mechanisms between a UAC and other SIP entities including UAS, proxy, and registrar. Information about the use of headers in relation to SIP methods and proxy processing is summarized in Table 1.

| Header field | where | proxy | ACK | BYE | CAN | INV | OPT | REG |
|---|---|---|---|---|---|---|---|---|
| Security-Client | R | ard | - | o | - | o | o | o |
| Security-Server | 421,494 | | - | o | - | o | o | o |
| Security-Verify | R | ard | - | o | - | o | o | o |

| Header field | where | proxy | SUB | NOT | PRK | IFO | UPD | MSG |
|---|---|---|---|---|---|---|---|---|
| Security-Client | R | ard | o | o | - | o | o | o |
| Security-Server | 421,494 | | o | o | - | o | o | o |
| Security-Verify | R | ard | o | o | - | o | o | o |

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.

- 421, 494: A numerical value 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.

- d: A proxy can delete a header field value.

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

method:

- o: The header field is optional.

## 3. Backwards Compatibility

The use of this extension in a network interface is a matter of local policy. Different network interfaces may follow different policies, and consequently the use of this extension may be situational by nature. UA and server implementations MUST be configurable to operate with or without this extension.

A server that is configured to use this mechanism, may also accept requests from clients that use TLS based on the rules defined in [6]. Requests from clients that do not support this extension, and do not support TLS, can not be accepted. This obviously breaks interoperability with some SIP clients. Therefore, this extension should be used in environments where it is somehow ensured that every client implements this extension or is able to use TLS. This extension may also be used in environments where insecure communication is not acceptable if the option of not being able to communicate is also accepted.

## 4. Examples

The following examples illustrate the use of the mechanism defined above.

### 4.1. Client Initiated

A UA negotiates the security mechanism to be used with its outbound proxy without knowing beforehand which mechanisms the proxy supports. The OPTIONS method can be used here to request the security capabilities of the proxy. In this way, the security can be initiated even before the first INVITE is sent via the proxy.

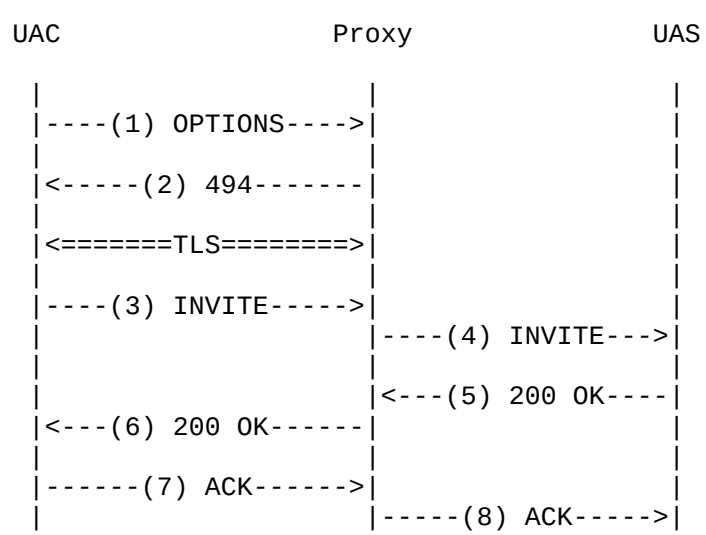

Figure 2: Negotiation initiated by the client

The UAC sends an OPTIONS request to its outbound proxy, indicating at the same time that it is able to negotiate security mechanisms and that it supports TLS and HTTP Digest (Step 1 of figure 1).

The outbound proxy responds to the UAC with its own list of security mechanisms – IPsec and TLS (Step 2 of figure 1). The only common security mechanism is TLS, so they establish a TLS connection between them (Step 3 of figure 1). When the connection is successfully established, the UAC sends an INVITE request over the TLS connection just established (Step 4 of figure 1). This INVITE contains the server's security list. The server verifies it, and since it matches its static list, it processes the INVITE and forwards it to the next hop.

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

More seriously, if the Security-Verify 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.

```
(1) OPTIONS sip:proxy.example.com SIP/2.0
    Security-Client: tls
    Security-Client: digest
    Require: sec-agree
    Proxy-Require: sec-agree

(2) SIP/2.0 494 Security Agreement Required
    Security-Server: ipsec-ike;q=0.1
    Security-Server: tls;q=0.2

(3) INVITE sip:proxy.example.com SIP/2.0
    Security-Verify: ipsec-ike;q=0.1
    Security-Verify: tls;q=0.2
    Route: sip:callee@domain.com
    Require: sec-agree
    Proxy-Require: sec-agree
```

The 200 OK response for the INVITE and the ACK are also sent over the TLS connection. The ACK (7) will contain the same Security-Verify header field as the INVITE (3).

## 4.2. Server Initiated

In this example of figure 3 the client sends an INVITE towards the callee using an outbound proxy. This INVITE does not contain any Require header field.

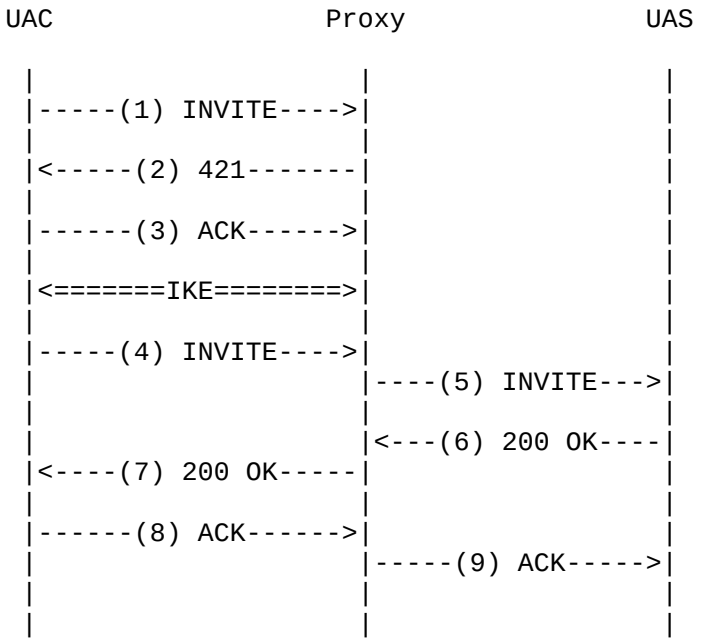


Figure 3: Server initiated security negotiation

The proxy, following its local policy, does not accept the INVITE. It returns a 421 (Extension Required) with a Security-Server header field that lists IPsec-IKE and TLS. Since the UAC supports IPsec-IKE it performs the key exchange and establishes a security association with the proxy.

The second INVITE (4) and the ACK (8) contain a Security-Verify header field that mirrors the Security-Server header field received in the 421. The INVITE (4), the 200 OK (7) and the ACK (8) are sent using the security association that has been established.

```
(1) INVITE sip:uas.example.com SIP/2.0

(2) SIP/2.0 421 Extension Required
    Security-Server: ipsec-ike;q=0.1
    Security-Server: tls;q=0.2

(4) INVITE sip:uas.example.com SIP/2.0
    Security-Verify: ipsec-ike;q=0.1
    Security-Verify: tls;q=0.2
```

## 5. 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 herein allow current networks using, for instance, HTTP Digest, to be securely upgraded to, for instance,

IPsec 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 and replay protection for the Security-Verify header field.

The security implications of this are subtle, but do have a fundamental 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 modifying 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 TLS [3]). In this situation, an insecure negotiation procedure allows attackers to trivially force even new devices to use only integrity protection.

Possible attacks against the security agreement include:

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.

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 of figure 1. 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.

Finally, attackers may also try to reply old security agreement messages. Each security mechanism must provide replay protection. In particular, HTTP Digest implementations should carefully utilize existing reply protection options such as including a time-stamp to the nonce parameter, and using nonce counters [4].

All clients that implement this specification MUST select HTTP Digest, TLS, IPsec, or any stronger method for the protection of the second request.

## 6. IANA Considerations

This specification defines three new header fields, namely Security-Client, Security-Server and Security-Verify that should be included in the registry for SIP header fields maintained by IANA.

This specification defines the 'sec-agree' SIP option tag which should be registered in IANA.

This specification also defines a new SIP status code, 494 (Security Agreement Required), which should be registered in IANA.

## 7. Acknowledgments

The authors wish to thank Lee Valerius, Allison Mankin, 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.

## 8. Normative References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler "SIP: Session Initiation Protocol", Work in Progress, draft-ietf-sip-rfc2543bis-09.txt, IETF, February 2002.

[2] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, IETF, November 1998.

[3] T. Dierks, C. Allen, "The TLS Protocol Version 1.0", RFC 2246, IETF January 1999.

[4] Franks, J. et al, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, IETF, June 1999.

[5] B. Ramsdell and Ed, "S/MIME version 3 message specification", RFC 2633, IETF, June 1999.

[6] H. Schulzrinne and J. Rosenberg, "SIP: Locating SIP servers", RFC 3263, IETF, June 2002.

[7] S. Kent and R. Atkinson, "IP Authentication Header", RFC 2402, IETF, November 1998.

[8] S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, IETF, November 1998.

[9] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, IETF, November 1998.

## 9. Non-Normative References

[10] M. Garcia, D. Mills, G. Bajko, G. Mayer, F. Derome, H. Shieh, A. Allen, S. Chotai, K. Drage, J. Bharatia, "3GPP requirements on SIP", draft-garcia-sipping-3gpp-reqs-00.txt. Work In Progress, IETF, October 2001.

## 10. Authors’s Addresses

Jari Arkko
Ericsson
02420 Jorvas
Finland
EMail: Jari.Arkko@ericsson.com

Vesa Torvinen
Ericsson
02420 Jorvas
Finland
EMail: Vesa.Torvinen@ericsson.fi

Gonzalo Camarillo
Ericsson
02420 Jorvas
Finland
EMail: Gonzalo.Camarillo@ericsson.com

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