Internet Engineering Task Force
Internet Draft

Sriram Parameswar
Brian Stucker
Nortel Networks

March 2002
Expires August 2002
<draft-spbs-sip-negotiate-01.txt>

The SIP Negotiate Method

Status of this Memo

Abstract

There is a need to negotiate a multitude of parameters, settings, and algorithms when setting up sessions using Session Initiated Protocol (SIP). While SIP itself provides mechanisms for negotiation of these parameters on a per-session basis through the use of the INVITE method, it does not provide a ready mechanism for meta-session negotiation. The closest mechanism provided is the REGISTER method, however, this method is directed towards the registrar alone, and cannot be used to conduct negotiation of parameters between any two arbitrary SIP nodes.

Examples of parameters that may need to be negotiated (and thus, the ready impetus for providing a simple mechanism to handle this) include: compression algorithms, code book size, message integrity

mechanisms, encryption algorithms, etc. Many of these parameters are not always eligible for use in an INVITE method, for two reasons:

- The INVITE method describes the parameters for initiation of a particular session. Once the session is over, the negotiated settings (such as the RTP profile used, in the case of SDP) are invalidated for future re-use. It would be inefficient to have to renegotiate parameters that are the same from session to session, or have to transmit large quantities of persistent data (such as a code book) each time.

- Many meta-session applications (and therefore their attendant negotiable parameters) are best utilized if they can be applied for the first message of a session.
  An example of this would be header compression. If the INVITE were compressed, then the header that identifies the type of compression in use would also be compressed, and therefore unintelligible (assuming no shim mechanism).

This document seeks to solve these problems by introducing a SIP extension that allows for meta-session parameters to be negotiated in a generic manner. This negotiation would take place prior to session establishment, between any two SIP entities (User Agents, Proxies etc.).

1 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant SIP guidelines

2 Introduction

SIP sets up multimedia sessions between User Agents. Given the complexity of these sessions, and the variety of User Agents and their capabilities, there are several session parameters that need

to be pre-defined or negotiated prior to establishing these sessions.

There is a real need to have a generic mechanism that will help negotiate these parameters a-priori. The initial goal was to establish a generic approach to negotiation, which would be applicable across all protocols used in the Internet world. Given the difficulty in coming up with a sufficiently generic approach that would be acceptable, the authors decided to attack a more immediate problem in the SIP arena and establish a generic negotiation mechanism.

This draft does not describe an extension which may be used

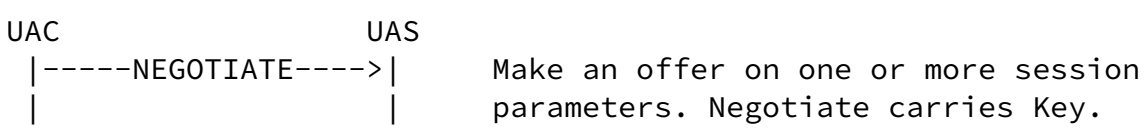
directly; it must be extended by other drafts by defining payloads (preferably XML based) to perform the actual negotiation.

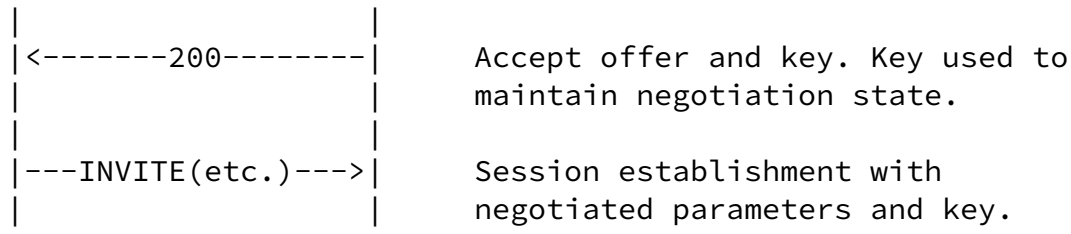
The result of the negotiation is a key that will be used in subsequent transactions to maintain negotiation state. This key is carried either as a shim i.e. between the transport and application layers (SIP and UDP for example) or as a SIP header (Key) in further SIP messages.

This new method may be used in both end-to-end and hop-by-hop scenarios.

2.1. Overview of Operation

In general the network entity that needs to negotiate meta-session parameters sends a NEGOTIATE with the payload indicating the offered parameters/algorithms.
The NEGOTIATE follows the offer-answer model as described in [4], the entity that offers the parameters must be willing and able to support all parameters in the offer.
The offer is placed in a NEGOTIATE payload. The offer may be accepted or rejected, in case the offer is accepted the answer is returned in the payload of a 200 OK. In case of a rejection a 488 or 606 is returned.





Negotiations expire and must be refreshed in exactly the same manner as registrations (see RFC 2543 [1]).

3.0 NEGOTIATE Method

"NEGOTIATE" is added to the definition of the element "Method" in the SIP message grammar.

The NEGOTIATE method is used to set up session parameters and algorithms prior to session establishment.

As with other methods the NEGOTIATE method name is case sensitive.

This document specifies the named extension 'negotiate'.

This feature name is placed in the Proxy-Require, Require, or Supported header in requests; and the Require, Supported, or Unsupported header in responses.

Header	Where	NEGOTIATE
Accept	R	o
Accept	2xx	o
Accept	415	c
Accept-Encoding	R	o
Accept-Encoding	2xx	o
Accept-Encoding	415	c
Accept-Language	R	o
Accept-Language	2xx	o
Accept-Language	415	c
Alert-Info	R	o
Alert-Info	180	-
Allow	R	o

Allow	2xx	o
Allow	405	m
Allow	r	o
Authentication-Info	2xx	o
Authorization	R	o
Call-ID	gc	m
Call-Info	g	o
Contact	R	m
Contact	1xx	-
Contact	2xx	m
Contact	3xx	-
Contact	4xx	-
Contact	5xx	-
Contact	6xx	-
Content-Encoding	e	o
Content-Length	e	o
Content-Type	e	*
CSeq	gc	m
Date	g	o
Error-Info	300-699	o
Encryption	g	o
Expires	g	o
From	gc	m
Hide	R	o
In-Reply-To	R	-
Max-Forwards	R	o
Min-Expires	423	m
MIME-Version	g	o
Organization	g	o
Priority	R	o
Proxy-Authenticate	407	o
Proxy-Authorization	R	o
Proxy-Require	R	o
Require	R	o

Table 1 Summary of header fields, A-R

Header	Where	NEGOTIATE
Retry-After	R	-
Retry-After	404,480,486	o
Retry-After	503	o
Retry-After	600,603	o
Response-Key	R	o

Record-Route	R	o
Route	R	-
Server	r	o
Subject	R	o
Supported	g	o
Timestamp	g	o
To	gc(1)	m
Unsupported	420	o
User-Agent	g	o
Via	gc(2)	m
Warning	r	o
WWW-Authenticate	401	o

Table 2 Summary of header fields, R-Z

3.1 Header Field Support for NEGOTIATE Method

Tables 1 and 2 add a column to tables 4 and 5 in the [rfc2543]. Refer to Section 6 of [1] for a description of the content of the tables. Note that the rules defined in the enc. and e-e columns in tables 4 and 5 in [1] also apply to use of the headers in the NEGOTIATE request and responses to the NEGOTIATE request.

3.2 Responses to the NEGOTIATE Request Method

If a server receives an NEGOTIATE request it MUST send a final response.

A 200 OK response MUST be sent by a UAS for an NEGOTIATE request that is successful. The 200 OK MUST carry the key sent in the original NEGOTIATE (see section 6.0) that is used to maintain negotiation state.

4.0 New Headers

This table expands on tables 4 and 5 in RFC 2543 [1], by adding support for the SUBSCRIBE, NOTIFY and NEGOTIATE.

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG	SUB	NOT	NEG
Key	g		o	o	o	o	o	o	o	o	m

4.1 "Key" header

The following header is defined for the purposes of this specification.

```
Key             =  [( "Key" | "y" )
                   (":" key-format)
                   ("=" key-param)]
key-param       =  1*( alphanum)
key-format      =  ( "Header" | "Key32" | "Key8" )
```

Examples:

Key: Header=ab45c6d2811e4681

y: Key32=1e6f9a33

Key: Key8=2c

4.1.1 Semantics of the Key Header

The role of the Key header is critical to creating a meta-session parameter sets that reflect the outcome of negotiations between various end-points. It is used to identify the resultant set of parameters stored at the endpoints after a completed negotiation.

Because SIP does not provide a generic, non-registration, method of referring to parameters that are kept and used across multiple sessions; identification of these parameters is requires an extension to the existing SIP specification in order to work under all circumstances. This becomes especially important when parameters cause the general headers of the SIP message to be unintelligible prior to some type of decoding (ie. decompression, decryption, etc.).

Additionally, it may be important that a SIP message is routable without exposing the entire contents of the message itself. This was identified in SIP. However this requirement may need to take a back seat to other considerations, for example bandwidth efficiency concerns, where the contents of the entire message may not be intelligible. As you can see, this becomes very tricky to identify which set of meta-session parameters to use to handle any given message between two endpoints.

As a result, we offer two mechanisms in order to provide a means by which to transport the Key information: header format, and shim format. The key is always transported in header format, as

a normal, general SIP header when present in a NEGOTIATE method transaction. Thereafter, it may be transported in either format as indicated by in the original NEGOTIATE method.

The key8 is an 8 bit Key and the key32 is a 32 bit Key, and either maybe transported in a subsequent shim structure. The shims are carried between the transport protocol (UDP, SCTP etc.) and SIP. The use of the shim allows the entire SIP message to be treated in the negotiated manner for example: the entire message may be encrypted or compressed. The shim allows for the recipient to perform the requisite decoding based on the negotiated algorithm.

Shims work well when they are negotiated hop-by-hop. However, there exists a need to transit through a proxy and expose only enough information for it to route on the message. In such situations the header format is used. The header format places the Key in clear text in a SIP message with any other pertinent information required by a proxy in the clear. Thus end-to-end negotiations may be performed in networks with intervening proxy hops.

5.0 Key Generation

Keys must be relatively unique, and therefore must be generated using an algorithm with distinct enough inputs to ensure that a 1-to-1 relationship exists between any single endpoint and the key that maps to that endpoint. Keys are also symmetric. Meaning that both endpoints use the same key value to describe the other endpoint.

One way of ensuring this is to use an MD5 hash of values pertaining to both the parties of a NEGOTIATE message. This could then be used in the case that the key is transported in a header, or distilled to an acceptable size for the shim in use. In order to provide a consistent means of generating keys, implementations SHOULD use the following algorithm:

> The TO URL, FROM URL, Call-ID value, and Via Branch value (if no branch is present, then an alphanumeric 0 should be used instead) should be stripped of all whitespace, and then concatenated together to form a

single string. This string should have the MD5 hash algorithm applied to it, to create the negotiated key. This is the key that is transported as part of the NEGOTIATE message.

The same callid should NOT be reused between two given endpoints when sending NEGOTIATEs, in order to reduce the likelihood that the same MD5 hash is generated. If an endpoint receives a NEGOTIATE with a key value that already exists, it should reject the request (with 400 "Bad Request" or 409 "Conflict").

Example:

```
NEGOTIATE sip:broker@example.com SIP/2.0
Via: SIP/2.0/UDP client.example.com:5060
From: Endpoint <sip:user@client.example.com>;tag=88a7s
To: sip:broker@example.com
Call-ID: 3248543@client.example.com
CSeq: 1 NEGOTIATE
Content-Type: application/xpidf+xml
Content-Length: 120
...
```

Starter String fed in MD5:

```
SIP:Broker@example.comENDPOINT<sip:user@client.example
.com>;TAG=88A7S3248543@client.example.com0
```

MD5 Key output (representation):

ab45c6d2811e4681f23513

5.1 Conversion to Key8

When Key8 is used, the negotiated key must be distilled into an 8-bit value in order to fit into the shim properly (thus the term Key8). This may be done by selecting the 8 most significant bits of the negotiated key. Note, that this format should only be used between a VERY small set of user agents, as the potential for key

collision, and subsequent need to recalculate a new callid, and therefore a new key to try the negotiation again, increases.

5.2 Conversion to Key32

When Key32 is used, the negotiated key must be distilled into a 32-bit value in order to fit into the shim properly (thus the term Key32). This may be done by selecting the 32 most significant bits of the negotiated key (in network byte format). In practice a random 32 bit number is likely to yield similar results.

6.0 Key Transport

6.1 Key Transport By Header

When the Key has been setup to be transported via a general header, both endpoints continue to send a valid, clear-text, SIP start line followed immediately by the Key header, which is also sent as clear text. More of the headers of the message may be in the clear, but at a minimum these two MUST be readable.

This is done so that intermediary proxies may still be able to route the request based off of the request URI in the start line, and still be able to maintain some semblance of context by using the Key header itself. Since the message data may not be readable to such proxies, it should use the Key header alone to keep track of context. This will likely not give enough information for the proxy to be able to operate if it is call-stated, in which case a hop-by-hop negotiation should be used so the proxy is able to operate more completely. Note, that since only the Key header form of transport has a notion of endpoint context, that the shim should not be used to keep context alone.

It is recommended to allow the same headers used in the construction of a key (TO, FROM, CALLID, and the top VIA), plus the CSEQ to be in the clear in addition to the fields above if an intermediate proxy is used. This is due to the fact that the key is only sufficient to keep context in terms of a single request/response pair, and is very limited in that regard for use at a proxy.

When meta-session parameters which exhibit hop-by-hop nature are to be supported - it is strongly recommended that intermediate proxies insert themselves into the NEGOTIATE request process (acting like a back-to-back UA) in order to ensure that all requirements of the proxy are satisfied regardless of the meta-session settings. This would require proxies to be cognizant of the NEGOTIATE method and its payload.

6.2 Key Transport By Shim

The key, as stated before, may be sent as part of a shim to the SIP protocol. This is offered in order to help minimize the overhead to the message itself, particularly for environments where every byte counts, and heavy compression is desirable. This format also allows the entire message to have the meta-session parameters applied to it, which can be useful in cases where encryption is desired and all of the headers of the SIP message must be included.

Additionally, keys must be prepended by a start byte. This is so a key is not confused with a normal start character for any given sip message. As a result, the top 6 bits of the first byte are zeroed out so that no valid ASCII character may result, and the shim can be detected by way of an illegal character.

6.2.1 Key32 Shim Format

The following shows the format of the Key32 shim (in which the 32-bit key is prepended by a shim start byte). Note that this format is what is actually placed in all messages post-negotiation

i.e. 32-bit key with start byte.

Byte 0: (MSB->LSB) [0000 00XY]

X = Version bit: Should always be set to zero (0)
Y = Shim Format: 0 = Key32, 1 = Key8.

Byte 1..4 (key)

Example:

0:0000 0000
1:1010 1011
2:0100 0101
3:1100 0110
4:1101 0010

Would be the 5 byte Shim32 representation of the key:

ab45c6d2

6.2.2 Key8 Format

The following shows the format of the Key8 shim (in which the 8-bit key is prepended by a shim start byte). Note that this format is what is actually placed in all messages post-negotiation i.e. 38-bit key with start byte.

Byte 0: (MSB->LSB) [0000 00XY]

X = Version bit: Should always be set to zero (0)
Y = Shim Format: 0 = Key32, 1 = Key8.

Byte 1 (key)

Example:
0:0000 0001
1:1010 1011

Would be the 2 byte Shim8 representation of the key:

ab

7.0 NEGOTIATE operation

This section deals with the specifics of how the negotiation mechanism works.

NEGOTIATE is suitable for use between any two SIP entities i.e they may be used between User Agents, SIP Proxies, BBUAs in any

combination. Examples include UA to Proxy, Proxy to Proxy, UA to

UA, UA to BBUA etc.

There are no restrictions on when a NEGOTIATE method is sent, however the authors recommend that if it is used in the middle of an existing session that the resultant key and parameters NOT be used in the context of the existing session. They may be used from the next session onwards.

7.1 Message Body Inclusion

The NEGOTIATE method requires a Message body to carry meaningful meta-session information. The answer is carried in the 200OK as a similar message body. With this in mind the authors have make the following recommendations:

(1) The NEGOTIATE/200 OK payload is in XML.
(2) Multiple NEGOTIATE payloads are permissible in a single NEGOTIATE, they are enclosed using Multipart MIME.
(3) Body contents are not changed/manipulated by intermediate proxies.
(4) Use of S-MIME for added security is permitted.

7.2 Negotiation Duration

It is recommended that the Expires header be used to impose a limit on the duration of a negotiation. The use of a Key after the expiry of the imposed time limit will result in a 4xx response (488 recommended). On expiry of the negotiation time limit, the expired KEY is removed from the list of KEYs maintained.

7.3 Terminating a Negotiation

Either side may terminate a negotiation at any time. The termination is achieved by sending a NEGOTIATE with the "Expires" header set to "0." A successful termination will result in a 200 OK from the other side. In case of Negotiation termination both the request and response do not contain a message body. When a negotiation is terminated that KEY is removed from the list of KEYs maintained.

7.3 Refreshing of Negotiations

A negotiation may be refreshed at any time by the use of the re-NEGOTIATE mechanism. As with the re-INVITE, it is RECOMMENDED that re-NEGOTIATE be used only if the negotiated parameters need to be changed or prior to expiry. The final expiration time is placed in the Expires header in the response. Note that the receiver of a NEGOTIATE may decrease the expiry time
- as in rfc2543bis-08 section 10.3.

The receiving party MAY reject the negotiation with a response of 423 (Registration Too Brief). This response MUST contain a Min-Expires header field that states the minimum expiration interval the receiver is willing to honor.
In general the behavior of NEGOTIATE shall follow that of REGISTER in the rfc2543-bis08 section 10.3.

If no refresh for a negotiation - as identified by the KEY is received before its expiration time, that KEY is removed from the list of KEYs maintained.

An interesting point is whether re-NEGOTIATE is performed using the already negotiated parameters or in clear text, for example re-NEGOTIATE compressed with the negotiated compression algorithm. It is our recommendation that a re-NEGOTIATE be treated as a brand new NEGOTIATION (especially since it may be used to change existing parameters) and be done in clear text.

8.0 Behavior of SIP User Agents

Depending on the applications requiring the negotiation of a meta-session parameters and the implementation details of the UAC, the type transport used could be either Shim or Header (or both). However, a UAC MUST support the Key header transport format if it is received.

The particular format of the key should be placed into the message prior to sending. If a key is received that is not understood by the recipient (maybe it never negotiated that key), an appropriate 400 class response should be returned to the request so that the sender of the key can determine that the key is no longer valid for use.

Additionally, because the NEGOTIATE has an expiry timer associated with it, a UA MUST continue to allow the key to be used if a transaction began while the key was still good, however, new transactions MAY not be allowed to use the key during this period.

8.1 Behavior of SIP Proxy and Redirect Servers

8.1.1 Proxy Server

Unless stated otherwise, the protocol rules for the NEGOTIATE request at a proxy are identical to those for a BYE request as specified in [1].

8.1.1.1 Stateless Proxy Server

The case where a stateless proxy server is involved can cause problems. In particular, the statelessness of the proxy precludes it from creating meta-session parameters since it has no notion of transaction, much less session. This means that meta-session parameters that cause the SIP message headers themselves to become unroutable by a stateless proxy should be used with care. The only way around this problem is to explicitly state all of the meta-session parameters by way of some header that is always readable by the stateless proxy. For example, in cases where algorithms such as dynamic compression are employed, this would involve sending the entire dictionary along with each message, thereby defeating the whole purpose of the compression. For other meta-session parameters, this may not pose a similar problem. However, since this draft seeks to simply provide a method of negotiation, and does not seek to recommend the particulars of what is negotiated; it is left up to the implementor to decide what is appropriate for their particular needs.

It is recommended, that where a stateless proxy is known to likely exist, that the key be transported via the header mechanism in order to better ensure that the stateless proxy is able to correctly route the SIP message (assumes that routing information is left readable by the stateless proxy).

8.1.1.2 Transparency of the Negotiation

An interesting case exists where a proxy wishes to view information in subsequent session transactions, but may not support the NEGOTIATE method, or it's content.

For example, the proxy should forward the NEGOTIATE, even though it does not understand it, according to RFC 2543. As a result of the NEGOTIATE, the two UA's decide to compress their SDP content. Later, an INVITE hits the same proxy, only this time the

SDP is unreadable due to the meta-session parameters setup as part of the NEGOTIATE. In this case, it may not be able to do it's job (billing, for instance) because the NEGOTIATE was transparent, but the effects of the negotiation are not. In this case, the proxy should not act any differently than it normally would if an INVITE came in with malformed SDP.

If a proxy cannot transparently process SIP transactions that are affected by meta-session parameters, they should not simply pass along a NEGOTIATE, and instead assert itself in order to ensure that SIP messages passing through it can be processed without interruption.

One method of doing this would be to act as a back to back user agent in respect to the NEGOTIATE and it's meta-session parameters. Another would be to intercept, and reject a NEGOTIATE that contains parameters that are not acceptable to the proxy. A proxy, however, MUST never alter the contents of the NEGOTIATE without causing a new Key header to be generated.

8.1.2 Forking Proxy Server

In the case where a forking proxy server is involved in the signaling path, such as sending a NEGOTIATE to a user's public address, which is resolved into a contact list with several entries, there exists the possibility for confusion on how to apply the NEGOTIATE across the various contacts.

In particular, depending on the application that is using the NEGOTIATE to set it's meta-session parameters, the application may not be able to cope with multiple endpoints using the same key.

An example of this would be using dynamic compression where each entity keeping a dictionary to use for compression is not aware of the dictionaries that are being update on peer branches. Because of this, the originator of the NEGOTIATE would wind up having to deal with two separate dynamic dictionaries that are guaranteed to never be synchronized, causing compression to intermittently fail.

Another example where no problem exists with forking a NEGOTIATE would be where a static algorithm is being NEGOTIATED, and application of that algorithm is entirely optional (thus not every branch of the NEGOTIATE need accept the negotiation parameters). This could easily be handled. An example of this would be the use of a static, well-known, compression algorithm.

Because the problems with forking vary according to the application that is employing the NEGOTIATE mechanism, it is recommended that the UAC that initiates the NEGOTIATE request be aware of it's specifics needs in this regard. If multiple responses are received, indicating that a proxy forked somewhere, and the UAC's application cannot handle this, the UAC SHOULD send another NEGOTIATE to the contacts that it wishes to cancel the unintended negotiation with the same key value used in the original NEGOTIATE, and an expiry value of zero.

UAC's SHOULD NOT send a CANCEL in order to cause a negotiation to be terminated, however UAS's MUST be prepared to receive a CANCEL for a NEGOTIATE, and handle it as if it were a NEGOTIATE with an expiry value of zero.

8.1.3 Redirection Server

Unless stated otherwise, the protocol rules for the NEGOTIATE request at a proxy are identical to those for a BYE request as specified in [1].

9. Guidelines for extensions making use of NEGOTIATE

It is strongly recommended that prior to designing XML payloads to use NEGOTIATE, the implementors ensure that a similar function cannot be performed using existing SIP mechanisms.

The use of NEGOTIATE may be for uni-directional or bi-directional meta-session information. For example using SCRIBE compression from UAC to UAS only, and UAS chooses to use ROGER towards the UAC. The XML payload SHOULD allow for negotiation of directionality.

10. Security Considerations

The worst damage that a mischievous intermediate could do would be to intercept, and handle a NEGOTIATE as if it were the intended recipient in order to gain access to information that it might otherwise not have (encryption key, etc.). For this reason, implementors should take care in how, and what they choose to send in a NEGOTIATE or its responses.

Sending an 8-bit encryption key in a NEGOTIATE to be used to encrypt sensitive information over the SIP signaling path for the next 10 years, for instance, would be a poor implementation.

In general, the security precautions used to guard against attacks against the sessions that the NEGOTIATE's meta-session parameters will apply to SHOULD be applied to the actual NEGOTIATE itself. Use of S-MIME for message body is encouraged.

Other types of attacks must also be taken into consideration, such as an intermediate tampering with the key value (causing sessions to be potentially compromised), or with the contents of the NEGOTIATE itself (causing confusion between the endpoints as to the value of the meta-session parameters). These, however are no worse than an intermediary scrambling any given piece of any SIP message, and are therefore, not unique outside of the amount of time such an attack may cause problems.

OPEN ISSUE: Should a negotiation be authenticated first? That is should the reply be a 407 with challenge and negotiation proceeds post authentication? This is interesting because the NEGOTIATE could be used to negotiate authentication mechanisms themselves.

10.1 Message integrity and authenticity / Man-in-the-middle Attacks

One of the intended uses for the negotiation mechanism is to set up authentication and message integrity algorithms. Thus the NEGOTIATE method itself is not subject to these protections - this makes the recipient vulnerable to the Man-in-the-middle attacks. Views are solicited to mitigate this security consideration.

10.2 Denial of service attacks

The recipient of an offer is allowed to increase or decrease the Expires value of a NEGOTIATE. This provides a measure of protection against DOS attacks. The Min-SE header issue (see 7.3) is of relevance here and the authors look for comments in this area.

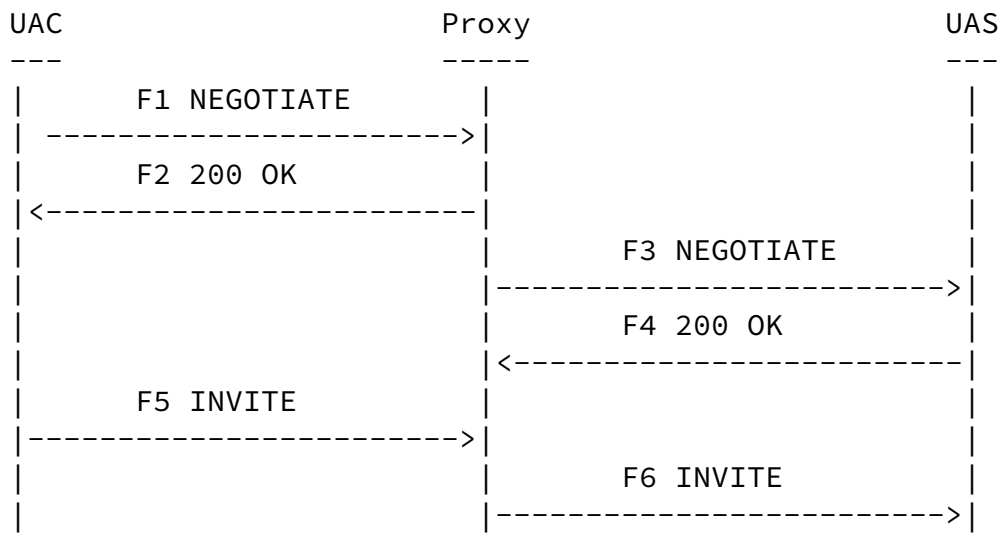
11.0 IANA Considerations

An option-tag "negotiate" is defined by this document. According to [1] must register any new option with the IANA. The authors intend to do this once the document has a measure of acceptance.

12 Example message flows

This section outlines some example flows to illustrate the use of the new NEGOTIATE method and Key header.

12.1 Hop-by-Hop Negotiation



In the scenario above the UAC wants to negotiate with an intervening Proxy server, which is typically stateful or a BBUA. The above also has applicability in wireless networks where the mobile negotiates several meta-session parameters with the Proxy Call State/Session Control Function (P-CSCF). It must be noted

that the XML documents carried in the messages below are just an illustration, and the authors expect XML documents to be defined that will suit the negotiation process.

In the messages the UAC offers ROGER and SCRIBE as possible compression mechanisms. The Proxy responds with a 200 OK and picks SCRIBE. The 200 OK returns the Key sent by the UAC, thus confirming that the next set of messages may be compressed. Also note that the negotiation from the Proxy to the UAS is not shown for brevity.

F1 NEGOTIATE UAC->Proxy

```
   NEGOTIATE sip:proxy.visited.example.com SIP/2.0
   Via: SIP/2.0/UDP bobmobile.example.com:5060
   From: Bob <sip:bob@mobile.example.com>;tag=88a7s
   To: Proxy <sip:proxy.visited.example.com>
   Key: Header=ab13984bdef
   Expires: Fri, 01 Jan 2010 16:00:00 EST
   Call-ID: 3248543@bobmobile.example.com
   CSeq: 1 NEGOTIATE
   Content-Type: application/xml
   Content-Length: 120

   <?xml version="1.0"?>
   <negotiate meta-session-info="compression">
     <tuple entity="sip:bob@mobile.example.com" />
     <tuple qualifier="ROGER" qualifier="SCRIBE" />
   </negotiate>
```

F2 200 OK Proxy->UAC

```
   SIP/2.0 200 OK
   Via: SIP/2.0/UDP proxy.visited.example.com:5060
   From: Bob <sip:bob@mobile.example.com>;tag=88a7s
   To: Proxy <sip:proxy.visited.example.com>;tag=8321234356
   Key: Header=ab13984bdef
   Expires: Fri, 01 Jan 2010 16:00:00 EST
   Call-ID: 3248543@bobmobile.example.com
   CSeq: 1 NEGOTIATE
   Contact: <sip:proxy@110.111.112.113>
   Content-Type: application/xml
   Content-Length: 95

   <?xml version="1.0"?>
   <negotiate meta-session-info="compression">
     <tuple entity="sip:bob@mobile.example.com" />
     <tuple qualifier="SCRIBE" />
   </negotiate>
```

F3 and F4 are similar to F1 and F2 respectively and thus not reproduced. All content after Key in the INVITE (F5) is treated

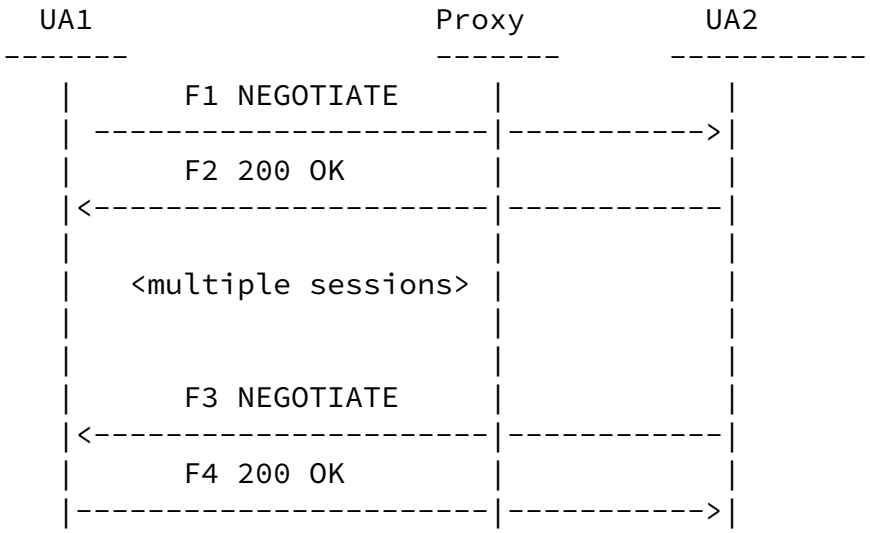
as compressed and thus not readable.

F5 INVITE UAC->Proxy

```
INVITE sip:joe.example.com SIP/2.0
Key: Header=ab13984bdef
a14adbe84585abe878693956cde5995affc
994adbeccff9494fa91344ffcdefac9474b
......
......
```

F6 is not shown.

12.2 End-to-End Negotiation and termination of negotiation



This example shows an end to end negotiation, with F1 and F2 same as above. The example further goes on to show how the UA2 can terminate the negotiation by sending the NOTIFY with an Expires value of zero. Note that in this case the NOTIFY MUST carry the Key and is not carrying the XML payload.

F1 and F2 similar to above except that Proxy simply routes the methods between UA1 and UA2.

F3 NEGOTIATE UA2->UA1

```
NEGOTIATE sip:proxy.visited.example.com SIP/2.0
Via: SIP/2.0/UDP bobmobile.example.com:5060
```

```
From: Proxy <sip:proxy.visited.example.com>;tag=9345ab
To: Bob <sip:bob@mobile.example.com>
Key: Header; ab13984bdef
Expires: 0
Call-ID: 978585@proxy.visited.example.com
CSeq: 1 NEGOTIATE
Content-Length: 0
```


F4 200 OK Proxy->UAC

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy.visited.example.com:5060
From: Proxy <sip:proxy.visited.example.com>;tag=9345ab
To: Bob <sip:bob@mobile.example.com>;tag=83ac6786
Key: Header; ab13984bdef
Expires: 0
Call-ID: 978585@proxy.visited.example.com
CSeq: 1 NEGOTIATE
Contact: <sip:proxy@110.111.112.113>
Content-Length: 0
```

13. References

[1] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.

[2] J. Rosenberg et.al., "SIP: session initiation protocol" (RFC2543bis-08), Internet Draft, Internet Engineering Task Force, February 21, 2002. Work in progress

[3] S.Donovan,J.Rosenberg, "The SIP Session Timer", draft-ietf-sip-session-timer-08. Oct 6, 2001. Work in Progress

[4] J.Rosenberg, H.Schulzrinne, "An Offer/Answer Model with SDP", Internet Draft, Internet Engineering Task Force, February 21, 2002 Work in progress

14. Author's Address

Sriram Parameswar
Nortel Networks
2380 Performance Drive
Richardson, Texas 75083
USA

Email: sriramp@nortelnetworks.com

Brian Stucker
Nortel Networks
2380 Performance Drive
Richardson, Texas 75083
USA

Email: bstucker@nortelnetworks.com