3GPP TSG WG3 Security - S3#19 Tdoc S3-0100347
Newbury, Great Britain
4-6 July, 2001

Agenda Item:

Source: Ericsson

Title: Integrity protection for SIP signaling

Document for: Discussion/Decision

1 Scope and objectives

The scope of this document is an in-depth discussion around the integrity protection of the Session Initiation Protocol (SIP) [SIP]. This document discusses three optional solutions: Cryptographic Message Syntax (CMS) [CMS], IPsec ESP and algorithm considerations.

An example of how good a simple compressor can compress CMS is also given. It shall be noted that this is only an example and it can be further optimized. The example gives about 43% extra overhead compared with the theoretical minimum i.e. 43 bytes vs. 30 bytes.

The difference in overhead for CMS with compression and the overhead for IPSec (24 bytes) is therefore in the same order. When comparing these two it should be noted that the CMS integrity protection (in this contribution) uses a 160 bit MAC i.e. 20 bytes while IPSec uses a truncated SHA1 MAC i.e. 96 bit MAC or 12 bytes.

Ericsson proposes that SA3 adopt the CMS protection mechanism as a working assumption for aSIP and incorporate that in the TS33.203 draft.

2 Background

SIP does not offer appropriate security mechanism for integrity protection. It has been proposed that SA3 should NOT develop any new mechanisms because some existing ones could be applied and because 3GPP specific solutions will hinder the access independence. Furthermore, mobile terminals must already support several security mechanisms and algorithms. The ever-increasing number of security features will become a burden for terminals if existing ones can not be reused. Ericsson has studied several optional solutions for this problem, including S/MIME, CMS and IPsec ESP. Further analysis of optional solutions has been needed.

A working assumption on SIP integrity protection has been that AKA defined in R'99 shall be reused. This means that long-term secret key K used for SIP authentication is shared between USIM and home network. Shorter-term secret key used for integrity protection (IK) is created during the registration in the USIM and home network. In the roaming case, P-CSCF receives the needed integrity key from the home network. The problem to be solved has been how to use these shared keys to protect the integrity of SIP messages in effective and efficient way.

3 CMS

At the Phoenix meeting, Ericsson presented an illustration on how integrity protection using CMS would look like in SIP. The studied solution was based on the idea of additional SIP integrity headers, which carried a CMS packet within the SIP packet. CMS would have included a keyed hash (HMAC) calculated over the whole SIP message.

The main problem with this solution was the overhead it caused for the SIP signalling. A rough estimate of the overhead was 124 bytes. An overhead of this size was perceived as too big. This chapter demonstrates how the integrity protection overhead can be minimised using shorter SIP header fields and compression.

The size of the SIP integrity header can be minimised by replacing relatively long strings like 'Proxy-Integrity'

with shorter strings like ‘Proxy-Int’, or by removing information about the encoding method from the header (base64 encoding is normally used for CMS). The size of CMS packet is difficult to decrease because the Phoenix version of CMS was already highly optimised (Appendix 1).

The example SIP message in Appendix 2 demonstrates a further optimised integrity header. The total SIP integrity overhead using this example is totally of 110 bytes before compression (<LF> characters not counted).

Compression of SIP messages is probably needed even without SIP integrity headers. The effect of compression to the integrity protection overhead can be estimated in the following way:

Using the message in the annex we have the following SIP integrity header [text in ***blue*** is HMAC]:

> **Proxy-Int: CMS;** (=15bytes)
>
> **cmsp="MD0GCSqGSib3DQEJEDAwAgEAMCsGCCsGAQUFCAECMB8GCSqGSib3DQEHAQQU*JmoHldU0dHkbm2pcdObpetPa/eb*="** (=67 bytes+27bytes+1byte)

Now we just have to choose some appropriate compression algorithm in order to get concrete figures for our estimation. In the first case, we assume the use of static dictionaries and choose LZSS for algorithm. LZSS is a compression algorithm in the Lempel-Ziv family that can encode a reference to a string, which is maximum 18 bytes long with 17 bits.

The total length of the string above is 15+67+27+1bytes=110bytes=880bits. Let us skip the last character i.e. the “ and only treat 15+67+27 bytes = 109 bytes. The compression is based on the fact that strings have appeared previously in a message, sent and/or received. Text that is sent in clear text format is encoded with 9 bits per character. The HMAC usually can not be compressed and hence using LZSS this part will be slightly expanded. However the rest of the CMS string will be constant and can be compressed efficiently. Note that in the general case the first time the message is sent it might not be compressed much. However, using a static dictionary one could e.g. compress the **cmsp=”MD0GCSqG…** part etc.

If we assume the string and LZSS algorithm as described above we can do the following calculation:

The 15+67 = 82 bytes = 4*18+10 bytes can be compressed to 4*17 + 17 bits = 85 bits and the HMAC with 27 bytes will be expanded to 27*9 = 243 bits. In total this means that we have encoded 82+27 bytes = 109 bytes = 872 bits to 85 bits + 243 bits = 328bits, i.e. 41 bytes. This simple example shows a compression rate equal to 872/328=2.7 or about 60%.

In any case, theoretical minimum for compressed SIP integrity header is somewhere around 30 bytes (27 bytes for base64 encoded HMAC and a few bytes for the identifiers of the rest of the header which is always static). The effectiveness and efficiency of compression depends highly on used method.

Ericsson performed some preliminary compression tests in order to get some hands on experience on SIP compression. Tests were run using 16 different SIP messages with and without SIP integrity header (cf. Appendix 2). ROGER compression scheme was used [ROGER]. ROGER has been developed especially for reducing the problem which are caused from using ASCII protocols over bandwidth limited channels. ROGER is a stateful compression scheme, which is able to use information about previously processed data for further-compressing data under delivery. It is able to use static dictionaries for compression, which further improve the compression result. A static dictionary of SIP commands was used in these compression tests. Other similar compression schemes could have been used.

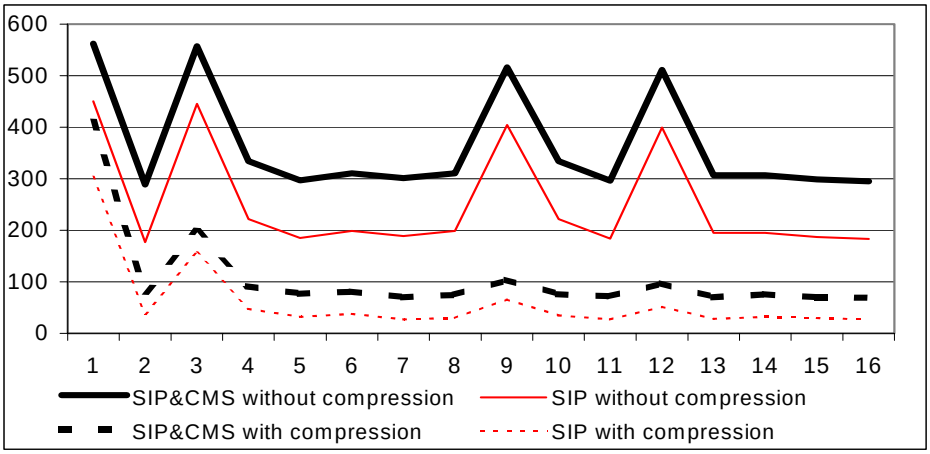

Figure 1: Compression of SIP messages. Lines from top to bottom are: 1) SIP with integrity header, 2) SIP without integrity header, 3) SIP with

integrity header compressed with ROGER, and 4) SIP without integrity header compressed with ROGER.

Figure 1 shows the test results. In the first throughput (cf. stateless compression), compression could downsize the SIP integrity overhead from 112 bytes into 106 bytes. In the following throughputs, the integrity overhead was downsized into 43 bytes. This is over 60% down from the original overhead. The effect of the compression to the SIP message alone is even better, over 80% down from the original size. Compression results could be improved by further developing the used dictionaries.

4 IPSec ESP

The main benefits of an IPsec-based approach would be very small overhead and the use of a well-defined existing mechanism. The overhead with the ESP protocol and a truncated SHA1 MAC (12bytes) was 24 bytes, cf. [S3010199]. The purpose of this part of a contribution is to discuss the use of IPsec for protecting SIP signalling. While many of the aspects – such as overhead and suitability for split UE networks – has been discussed in depth, some open questions have been discussed at the last meetings regarding how IPsec fits to the typical use of port numbers in SIP.

4.1. Typical SIP Port Behaviour

If the transport is a connection-oriented protocol (TCP or SCTP), a connection is opened when needed. All the requests and responses are sent through that connection. So, if the transport is connection-oriented, then the response is always sent through the same connection from where the request was received. Therefore, for connection-oriented transports we treat the whole connection with the same port numbers. In UDP, each “connection” is a single packet, and can (to an extent) use different port numbers.

In conclusion, the question we must ask when we consider the use of IPsec for the protection of SIP signalling during a registration is the following: how are the port numbers used in SIP (for both UDP and TCP), and what kind of IPsec policies are needed to protect these?

Let us first discuss the case of contacting a server. In this case, the server-side port number is always well known:

- Through the use of a port in the SIP URI.
- Certain types of DNS queries can also return a specific port for SIP.
- By default, the port is 5060.

Typically, the client-side port number is ephemeral. Note that SIP handles responses in a non-typical way; responses are not sent to the source port number but rather to the explicitly given IP address and port (which may be different from the actual source).

The standard does not dictate a particular implementation. It is common for a server to listen on port 5060 and send on a different port. It is common for a client to pick a dynamic port for sending (actually the OS does this) and then listen on port 5060. But none of this is mandatory. A dynamic port used by the client to send a message may be used just for that message, for the entire transaction, for the duration of a registration, or for the entire time that the client is "on". In general, running out of dynamic ports is not an issue.

4.2. Using IPsec with SIP

IPsec relies on protocol and port numbers to determine when to apply security. Some regularity must exist in the traffic flows in order for this to be possible; completely random use of port numbers would not allow separation of several clients behind one IP address. On the other hand, we wish to enable efficient P-CSCF implementations that can take advantage of multiprocessor systems and parallel processing.

In the following we propose a way for P-CSCFs to take advantage of efficient implementation schemes while allowing security to be run towards UEs.

1. We allow servers to use any port number for sending/receiving SIP packets. Typical OSes support such behaviour easily.

2. On the UE side, we require the SIP clients to use the same port for both sending and receiving. Typical OSes support such behaviour easily.

3. If there are multiple independent SIP clients in one UE – perhaps corresponding to several persons – they should use different ports of course.

With these rules it is sufficient to fully identify the SIP flows belonging to different UEs and the clients running inside them. For instance, if the P-CSCF is at the address P and the UE is at address U with two running clients on ports P1 and P2, then all traffic that runs on UDP from U:P1 to P:* has to be secured with the SA belonging to the first user.

5 Algorithm considerations

Ericsson promotes the reuse of existing security algorithms for SIP integrity protection. 3GPP specific solutions are not appropriate for integrity protection mainly because they hinder interoperability. It is important that algorithms used to protect SIP signalling are globally known and acceptable. Otherwise technical solutions may result in fragmented communication networks which are not interoperable in practice. Furthermore, the algorithm should be chosen from the set of standardised algorithms that will be necessary to implement in software in the terminal anyhow (e.g. in WAP). The implementation on the network side is not a critical issue. For these reasons, the algorithm should fulfil at least three requirements: recognition as a generally accepted algorithm, availability in handsets and efficiency as a software implementation. SHA-1 is very good candidate for such an algorithm.

Conclusions

This document has discussed on the following issues:

- Integrity protection using SIP integrity headers and CMS AuthenticatedData packets is efficient if compression is used. The overhead caused by the integrity header can be squeezed down close to 30 bytes (theoretical optimum). In practice, it has been demonstrated that overhead of 43 bytes can be reached by using ROGER compression scheme. At the same time, the size of SIP messages was decreased by 80%.

- The benefits of IPsec ESP include a well-defined and ready format that requires no further standardisation. Not full freedom can be given to how port numbers are used, but a reasonable solution was presented in section 4.2 which allows even split UEs and threaded/paralleled P-CSCF implementations.

- Algorithms used for integrity protection should be generally recognised in order to promote reusability and already available in handsets as efficient software implementations.

Ericsson still proposes to define the integrity protection mechanisms at SIP level mainly because of the ease of implementation due to reuse reasons, and because the same scheme could perhaps be used also for later end-to-end security in SIP. As an alternative to SIP level security IPSec-ESP with fixed policies is also acceptable. The difference in the bandwidth-efficiency of these mechanisms is not significant if compression is used.

Ericsson proposes that the integrity protection algorithm should be chosen from the set of standardized algorithms that are necessary to be included in handsets as software implementations. One good candidate for such an algorithm is SHA-1.

References

[CMS] Cryptographic Message Syntax, IETF, RFC 2630, June 1999.

[CMSCOM] P. Gutmann, Compressed Data Content Type for CMS, IETF, Internet Draft, draft-ietf-smime-compression-04.txt, May 8, 2001.

[DEFLATE] P. Deutsch, DEFLATE Compressed Data Format Specification version 1, IETF, RFC 1951, May 1996.

[ESP] IP Encapsulating Security Payload (ESP), IETF, RFC 2406, November 1998.

[ROGER] H. Hannu, J. Christoffersson & K. Svanbro, RObust GEneric message size Reduction (ROGER), IETF, Internet Draft, draft-hannu-rohc-roger-00.txt, February 2001.

[SIP] SIP: Session Initiation Protocol, IETF, Internet Draft, draft-ietf-sip-rfc2543bis-02.txt, November 24, 2000.

[ZLIB] P. Deutsch & J-L. Gailly, ZLIB Compressed Data Format Specification version 3.3, IETF, RFC 1950, May 1996.

[S3010199] Ericsson, Integrity protection for SIP signaling, Phoenix May 2001, 3GPP SA3#18

Appendix 1: Example of CMS packet construction

CMS AuthenticatedData format:

Field or Sequence name	**Content**	**Included / Excluded**
version	version	Included
recipientInfos	version keyIdentifier keyEncryptionAlgorithIdentifier encryptedKey	Excluded Excluded Excluded Excluded
macAlgorithmIdentifier	macAlgorithmIdentifier	Included
encapContentInfo	eContentTypeIdentifier eContent	Included Excluded
hmac	hmac	Included

BER-encoding:

CMS DATA	**BER-encoding**
SEQUENCE { OBJECT IDENTIFIER authenticatedData(1 2 840 113549 1 9 16)	00110000 00111101 00000110 00001001 00101010 10000110 01001000 10000110 11110111 00001101 00000001 00001001 00010000
SEQUENCE { version (INTEGER 0)	00110000 00110000 00000010 00000001 00000000
SEQUENCE { OBJECT IDENTIFIER macAlgorithm(1 3 6 1 5 5 8 1 2)	00110000 00101011 00000110 00001000 00101011 00000110 00000001 00000101 00000101 00001000 00000001 00000010
SEQUENCE { OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)	00110000 00011111 00000110 00001001 00101010 10000110 01001000 10000110 11110111 00001101 00000001 00000111 00000001
HMAC (OCTET STRING 2F 23 82 D2 F3 09 5F B8 0C 58 EB 4E 9D BF 89 9A 81 E5 75 4D)	00000100 00010100 00101111 00100011 10000010 11010010 11110011 00001001 01011111 10111000 00001100 01011000 11101011 01001110 10011101 10111111 10001001 10011010 10000001 11100101 01110101 01001101

Base64-encoding:

BER-encoding	**Base64-encoding**
00110000 00111101 00000110 00001001 00101010 10000110 01001000 10000110 11110111 00001101 00000001 00001001 00010000 00110000 00110000 00000010 00000001 00000000 00110000 00101011 00000110 00001000 00101011 00000110 00000001 00000101 00000101 00001000 00000001 00000010 00110000 00011111 00000110 00001001 00101010 10000110 01001000 10000110 11110111 00001101 00000001 00000111 00000001 00000100 00010100 00101111 00100011 10000010 11010010 11110011 00001001 01011111 10111000 00001100 01011000 11101011 01001110 10011101 10111111 10001001 10011010 10000001 11100101 01110101 01001101	MD0GCSqGSib3DQEJEDAwAgEAMCsGCCsGAQUFCAECMB8GCSqGSib3DQEHAQQUJmoHldU0dHkbm2pcdObpetPa/eb=

Appendix 2: Integrity protected SIP message using CMS (integrity header in bold)

```
INVITE sip:al@jaguar SIP/2.0
From: SIP:bo@e005004b57366:5061
To: sip:al@jaguar
Call-ID: e53cdd755e5decf@e005004b57366
CSeq: 1 INVITE
Subject: hello
Via: SIP/2.0/UDP e005004b57366:5061
Require: 100rel
Content-Type: application/sdp
Content-Length: 208
Proxy-Int: CMS;
cmsp="MD0GCSqGSib3DQEJEDAwAgEAMCsGCCsGAQUFCAECMB8GCSqGSib3DQEHAQQUJmoHldU0d
Hkbm2pcdObpetPa/eb="

v=0
o=bo 3177299996 3177299997 IN IP4 131.160.30.49
s=Internet Phone Call
c=IN IP4 131.160.30.49
t=3177299996 3177301796
m=audio 5006 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=ptime:20
```