**3GPP TSG SA WG3 Security — S3#16** **S3-000703**

**28-30 November, 2000**

**Sophia Antipolis, France**

**Source:** **Telenor**

**Title:** **Using HTTP to distribute MAPsec SAs**

**Document for:** **decision/discussion**

**Agenda Item:** **tbd**

## Using HTTP to distribute MAPsec SAs

This contribution adds a little substance to the idea of using HTTP as the transport protocol for distribution of MAPsec SAs.

The main idea, as discussed in this contribution, is to define the MAPsec SAs in terms of an XML documents/"homepages". XML is a very flexible text based data definition and representation language with meta-language capabilities. The natural way to express MAPsec SAs will then be to define a special MAPsec SA Document Type Definition (DTD). Actual MAPsec "homepages" will then be written within a MAPsec SA DTD context.

Retrieval of a MAPsec SA in this context will be done in the same manner as a browser fetches a homepage.

When using HTTP, one will basically have the option of either securing the connection on the network layer (IPsec) or on the transport layer (SSL/TLS).

## *NDS architecture sketch for MAPsec SA distribution*

The figure below shows how MAPsec SA distribution may be structured. A central root CA contains digital certificates for all operators. KAC-A and KAC-B retrieves certificates from the root CA as needed. KAC-A and KAC-B also operates as a local CA for its own network.

When KAC-A is fetching an SA from KAC-B the process is as follows:

- KAC-A attempts to access a MAPsec SA at KAC-B. The location of the MAPsec SA is defined by an URL.
- The access protocol is HTTP/1.1. It will be run over a TCP connection secured by TLS. The establishment of TLS requires a digital certificate, which can be retrieved from the root CA if not already available.
- The MAPsec SA, which is an XML document, is then transferred between KAC-B and KAC-A.

If MAPsec SAs are to be uni-directional, the procedure must be executed in both directions. Expiry of SAs can be supported directly in the HTTP/1.1 protocol or it can be explicitly defined in the SAs.

Basically, the same procedure for exchange of SAs between KACs can also be used for the KAC to MAP-NE case. The KAC will host a set of "homepages" containing SA/SA-pair for communication with all roaming partners. These pages will then be accessed by the MAP-NEs. For the KAC to MAP-NE case, SAs are transported in one direction only. If SAs are to be uni-directional, an SA-pair must be transferred.

Since SA synchronization and updating will not be immediate and effective in all nodes simultaneously, a scheme must be devised that allows for an overlap period where two different SAs will be valid for secured MAP communication between MAP-NEs. This overlap period should be specified in the SAs and an indicator must be provided to differentiate between the valid SAs.

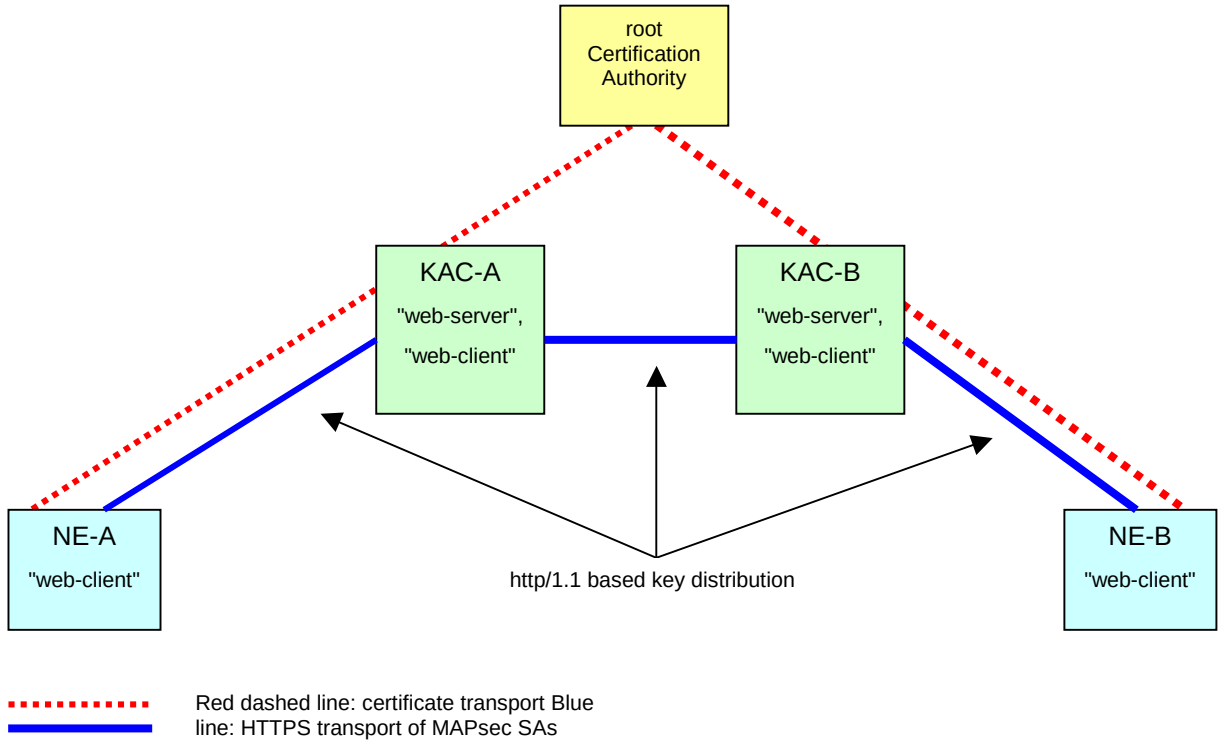


Figure-1: HTTP/1.1 based transport of MAP SA XML documents, secured by TLS

## *Expressing MAPsec SAs in terms of an XML Document Type Defintion*

XML[1,2,3] is a text based description and representation language designed for representing documents. Amongst the capabilities of XML is to define new Document Type Defintions (DTD). A DTD is a file (or several files to be used together), written in XML's Declaration Syntax, which contains a formal description of a particular type of document. It sets out what names can be used for element types, where they may occur, and how they all fit together. DTD's can be viewed as a type declaration for a document/homepage, and this makes it convenient to express MAPsec SAs in terms of "homepages" of a MAPsec SA DTD.

It is clearly possible to express MAPsec SAs in terms of XML documents with special MAPsec SA DTD. To have the SA defintion in a text representation is beneficial in terms of flexibility and ease of implementation. The only real drawback is the verbose nature of a text-based description, but this drawback will only be important if the total number and update frequency of MAPsec SA are very high. With the current assumption that SA-pairs are valid on a network-to-network basis, the verbosity of XML based MAPsec SAs is unlikely to become problem.

If the MAPsec SAs becomes too voluminous due to its text representation one may of course use compression techniques to reduce the document size. Support for compression is available in HTTP/1.1 [4],and GZIP[5], compress[1] and deflate[6] methods are pre-defined. Support for compression is also built into TLS/SSL.

[1] UNIX compress, using This format is an adaptive Lempel-Ziv-Welch coding (LZW).

### *Securing the HTTP/1.1 transport by means of SSL/TLS*

Security for HTTP/1.1can be provided at the network layer by means of IPsec. This contribution does not discuss that possibility, but this does not mean that the possibility of using of IPsec should be dismissed.

SSL[7] and TLS[8] is fairly similar and both provides security in a transport sub-layer just above TCP[2].

Basic security feature provided by SSL:

- The communicating parties can authenticate each other by using public key cryptography and digital certificates. The certificates to be used is based on X.509 version 3.
- Data above the transport layer is confidentiality protected by means of encryption. To the application layer the encryption is transparent. The initial session handshake and key negotiation allows for a separate compression stage to be included.
- Message authentication and integrity is provided by use of MACs. SSL uses a slightly modified version of the HMAC described in RFC-2104.

HTTP, when running on top of SSL or TLS, is often denoted HTTPS, and should be using port 443 instead of port 80 as would be the normal HTTP port.

The basic features provided by TLS is very much the same features that SSL proves. TLS version 1.0 is based directly on SSL version 3.0. The changes introduced with TLS mainly deals with removing support for proprietary (and unpublished) technologies and to become more aligned with methods used in IPsec RFCs[3]. Nevertheless, SSLv3 and TLSv1 is sufficiently different as to make interoperability difficult.

Since TLS is developed and published by a standards body and given that TLS is at least as good as SSL, it is recommended that only TLS is mandated.

Furthermore, should TLS be used in a UMTS NDS context it is recommended that only CAs that operate in a UMTS context should be used. It would probably be useful to have for instance GSM Association to operate as the root CA.

## *Concluding remarks*

This contribution discusses the possibility of specifying MAPsec SAs in XML and distributing MAPsec SAs by means of HTTP. For securing the transport of MAPsec SAs it is suggested to use TLS and to define a root CA within the UMTS operator context.

To develop a solution around the suggestions in this contribution is in my opinion feasible. The solution will be flexible in terms of how to define the MAPsec SAs. Furthermore, the use of existing technologies in their native ways should ensure that a working solution could be specified and completed fairly quickly.

One serious drawback with the solution suggested in this paper is that it requires TCP/IP support in MAP-NEs. This drawback will essentially apply to all MAPsec SA distribution solutions except for the case when MAP itself is used as transport mechanism.

---

[2] Neither SSL nor TLS supports UDP

[3] In particular on how to use HMACs

*References*


[1] The XML homepage: http://www.w3.org/XML/

[2] Extensible Markup Language (XML) 1.0 (Second Edition)
October 2000, http://www.w3.org/TR/REC-xml

[3] The XML FAQ
Version 1.6 (21 July 2000), http://www.ucc.ie/xml/

[4] RFC-2616: Hypertext Transfer Protocol -- HTTP/1.1
June 1999, ftp://ftp.ietf.org/rfc/rfc2616.txt

[5] RFC-1952: GZIP file format specification version 4.3
May 1996, ftp://ftp.ietf.org/rfc/rfc1952.txt

[6] RFC-1950: ZLIB Compressed Data Format Specification version 3.3
May 1996, ftp://ftp.ietf.org/rfc/rfc1950.txt

[7] The SSL Protocol, Version 3.0
March 1996, http://home.netscape.com/eng/ssl3/ssl-toc.html

[8] RFC-2246: The TLS Protocol, Version 1.0
January 1999, ftp://ftp.ietf.org/rfc/rfc2246.txt