3GPP TSG SA WG3 Security — S3#25 **S3-020534**

8 - 11 October 2002

Munich, Germany

Source: Ericsson

Title: MBMS - key management comparison

Document for: Discussion and approval

Date: 2-10-02

Scope

This discussion paper focuses on key-management protocols for MBMS. It provides some scenarios in which a key-management protocol for MBMS must work and properties such a protocol must have. The last part of the paper is a comparison of three candidates, and how these meet the requirements.

Introduction

Since MBMS will be aligned with IP-multicast as specified by IETF, it is natural also to look at the work done by the IETF working group MSEC when it comes to key-management. This discussion paper provides a comparison of the three key-management protocols specified by MSEC and their suitability for MBMS.

Abbreviations and Notation

These abbreviations are used through out the document. The list is provided for readers not familiar with security terminology. For definitions that are more exact and descriptions, we refer to the security literature and in particular www.ietf.org.

- *MBMS* Multicast Broadcast Multimedia Services
- *MB-SC* Multicast Broadcast Service Centre
- *IM* Identity module, e.g. SIM or USIM
- *IETF* Internet Engineering Task Force, a standardisation body
- *MSEC* Multicast Security, a working group in IETF
- *AA* Authentication and Authorisation
- *SA* Security Association parameters and keys for a secure session
- *PKI* Public Key Infrastructure, name of the infrastructure required for asymmetric cryptography
- *LKH* Logical Key Hierarchy, a family of methods for efficient re-keying of groups
- *IKE* an end-to-end key management protocol

- *SRTP, IPsec, TLS, SMIME* protocols for secure transmission of data
- *DH* Diffie-Hellman, a common method for deriving a shared secret
- *SHA-1, MD5* methods for computing a “fingerprint” of a message
- *MAC* Message Authentication Code, a commonly used name on the above mentioned “fingerprint”
- *DSA, DSS, RSA signatures, ECDSS* methods for digitally signing messages
- *CBC, OFB, counter mode* modes of operation for block ciphers
- *DES, 3DES, AES* block ciphers

The following notation is used:

- [x] means that x is optional
- ~ means approximately equal to

Scenarios

We are going to consider several scenarios through out the document, and in this section they will be described and numbered for future reference. It should be noted that the list of scenarios is in no way complete.

Scenario 1: The ten o’clock news

There is a show or media session that starts and ends at pre-determined hours. The media is multicast to all registered users, who can register for participation before or during the session. A user that registers after the start of the session will only be able to view the rest of the session (unless he has recorded all the data before joining). This is the “all in one” scenario, i.e. there is no re-keying done in this case.

Scenario 2: MTV-like video streaming

A MB-source streams music videos in an MTV-like fashion, and users can chose to join and leave the session at any time. This implies that when a user leaves, all the remaining users will have to be re-keyed. The re-keying is probably performed at regular intervals, and a user registers for a number of time-slots.

Scenario 3: Stock prices

A user wants to keep track of the prices of his stocks. He registers to an MB-source, whichmulticasts the prices every fifteen minutes. He probably signs up for a month or so in advance. In this case it is probably not necessary to re-key more than once every few days.

Requirements

This section discusses requirements put on a key-management protocol for MBMS.

Efficiency

Any key management protocol that has to deal with large groups must be very efficient in both the registration phase and during re-keying. There are two important types of scenarios to consider.

The first is the case where there are few users sporadically joining and leaving the group. A user joining the group is easily handled, but a single or a few users departing requires efficient re-keying of the remaining group members

(which might be a set of considerable size). Such re-keying is today handled best by LKH-schemes. In this case it is not desirable to use event-driven re-keying, but rather wait a certain time period, collect all users that leave during that period, and lock them all out in one batch.

The second type of scenario is that at the start of some MB-session there will be a lot of registration requests, e.g. at the beginning of the popular Friday movie. The key management protocol must be able to cope with a large set of registration requests in a short time span. The possibility that users may cancel their registrations before the session begins (effectively causing re-keying of all the remaining users) complicates things further. This must be handled in an efficient manner, not to open up for an easy DoS attack (and to handle normal de-registrations). A simplifying measure would be to disallow such de-registrations, but it might hurt business to do so.

Scalability

As mentioned in Section 3.1, the protocol must be able to handle the troublesome case with large groups in an efficient manner. Since the set of members of an MBMS-session is expected to be large point-to-point re-keying is not good enough. It implies poor utilization of network and computational resources. Instead, some kind of multicast re-keying is required.

Reliability

The re-keying must be error-tolerant to some extent. That is, just because a group member looses one or a few re-keying messages, it should be possible to regain synchronization. Since this property must rely on added redundancy in the message size or in the number of messages, the tolerance level must be weighted against the loss of efficiency the redundancy causes.

Key-Management Components

There are some fundamental design techniques that a key-management protocol relies on. Since some of them are quite costly, this section examines their need in the MBMS case.

Symmetric vs. Asymmetric Cryptography

Asymmetric cryptography is often viewed as a silver-bullet. But the price in terms of efficiency is sometimes too high; especially when thin clients are involved and servers are expected to serve a huge set of users within a short time frame.

An operator has a relation to its subscribers manifested by the subscriber key. The operator should take advantage of this relation, and use symmetric cryptographic algorithms to push the key to the privileged users. There is no immediate need for asymmetric cryptography in the case of MBMS, and the use thereof would impose an unnecessary load both on the clients and the MB-SC.

For the MB-SC there are scenarios where it is questionable whether asymmetric algorithms even would be feasible. Assume there is a set of 1 million users that wish to join a certain multicast session that starts at a given time. Then it is not unreasonable to expect all these users to join the group within a time frame of five minutes. This would lead to the server having to perform 1 million asymmetric key operations in five minutes. This would probably be a reasonable situation. According to www.mms.se the sports news show “Sportnytt” on Swedish television had 1,1 million viewers on 4/9 2002 (an ordinary Wednesday).

Is Re-keying Needed?

The key management can be separated in two parts, the registration phase and the re-keying phase. In the first phase a user asks the MB-SC for membership in a certain session. During this phase the user is also provided with the key currently used in the group. To exclude members the group key must be changed. This is done be providing the remaining group members with a new key which is then used.

Depending on the charging model used and the scenarios for the sessions, the re-keying might not be needed. The simplest case is that the users sign up and pays for one session, e.g. one TV-program. In this case the excluding of members, i.e. re-keying, is not needed. The session lasts for as long as the TV-program plays, after that the key is useless. Initially this is the most likely scenario. This hold under the assumption that once a user has registered for a session, de-registration is not possible.

Another scenario is the "MTV-scenario". Here a user might sign up and pay for 30 minutes of videos. This is different in the sense that there is no obvious start or end of the session, and users might join and depart sporadically. Here re-keying might be needed, unless the price for a 30-minute slot is so low that a user does not care if he pays for slot which he will not see in its entirety.

Evaluation of Existing Protocols

This section will evaluate some of the possible candidates for the key-management in MBMS. They are all under standardisation in the MSEC WG in the IETF.

Criteria for Evaluation

The protocols will be evaluated by examining how they perform according to the points listed below.

- **Number of round trips** describes the number of round trips needed for registration and for re-keying respectively.
- **Message sizes** describe the approximate size of the messages used for group management and key transfer.
- **Workload** describes the number of cryptographic operations performed by the client and the server respectively. Since asymmetric key operations today are by far more resource consuming than symmetric ones, the symmetric operations are not counted explicitly. In addition to cryptographic operations, there are of course also needs for message parsing/building, database lookups etc, but these operations are considered as negligible in comparison to asymmetric key operations.
- **Implementation footprint** describes the size of any available implementations. The implementations are measured in lines of code, but since none of them is complete, the measurements are not exactly comparable. However, they give a hint at the protocol sizes.
- **Suitability for different AA-frameworks** There are several frameworks for AA imaginable. Here we consider three possibilities. The first being the normal USIM/AuC authentication, the second an MBMS specific framework and the third a framework based on user certificates. In the case of USIM/AuC, a shared key is agreed upon by the use of AKA. If there is a specific AA-scheme constructed for MBMS, this will of course be made to fit the key-management decided upon. Hence an MBMS specific AA-framework is in the reminder assumed to fit any key-management protocol
- **Applicability to different security protocols** describes which protocols that can be used to secure the media session with information available after running the key-management protocol.
- **Algorithms used by the key-management protocol itself** describes which algorithms are used by the key-management protocol to handle the group, to which extent these can be substituted and what work is required to make such a substitution happen.
    - **Standardisation status** describes how far in the standardisation process the protocol has come.
    - **Security of protocol** describes what mechanisms are used to protect the key-management from some commonly known attacks.
    - **Validity for different scenarios** describes the suitability of the protocol for the different scenarios outlined in Section 2.
    - **Ease of deployment** describes how easy it is to integrate the protocol in the MBMS structure.

GDOI

Group Domain Of Interpretation [0] is both a registration and re-keying protocol, developed by Cisco, Verisign and Sparta.

Number of round trips

- Registration: 3 roundtrips (using IKE main mode phase 1), 1.5 roundtrips (using IKE aggressive mode phase 1). In addition to one of these exchanges a modified phase 2 with 2 roundtrips to obtain the first Data or Re-key SA is needed.
- Re-keying: 2 roundtrips or 0.5 roundtrips (push)

Message sizes

- Registration: between 1kB and 2kB (per message) plus a few hundred bytes for each message in the phase 2.
- Re-keying: 2 messages in each direction of a few hundred bytes each, followed by 2 more messages in each direction of a few hundred bytes plus optional certificates.

Workload

During the registration process the following operations are required:

- Server-side: 2-3 asymmetric key operations plus optionally one hash for integrity protection.
- Client-side: 2-3 asymmetric key operations plus optionally one hash for integrity protection and one additional asymmetric key operation to check the optional certificate.

During the re-keying phase the following operations are needed:

- Server-side: 1 DSS plus some symmetric block encryptions
- Client-side: 1 DSS check plus some symmetric block decryptions plus optionally one asymmetric key operation to check a certificate.

Implementation footprint

There is an open implementation of GDOI at http://www.vovida.org/ which has support for IPsec and SRTP under OpenBSD and Linux. It is not a full implementation, but consists of approximately 35000 lines of code.

Suitability for different AA frameworks

If GDOI is preceded by some AA protocol that is capable of deriving a shared key, the exchange will become a bit less expensive in the computational sense, since authentication can make use of the pre-shared key (derived using out of band signaling). The actual derivation of the key can however not benefit from any out of band derived key. Hence GDOI is partially suitable for USIM/AuC.

Since GDOI always performs authentication on its own, it is not really helped by any “stand alone” AA-protocol.

Applicability to different security protocols

GDOI only supports IPsec by default. Other protocols can be added by writing a new RFC to specify how it should be done. (Cisco has implemented support for SRTP in a prototype, suggesting that it is possible).

Algorithms used by protocol itself and their extensibility

- Confidentiality algorithms: DES, 3DES and AES (all in CBC mode)
- Hashes: SHA-1, MD5
- Signatures: RSA, DSS, ECDSS.

Other algorithms can be added, but requires a new RFC.

Status in standardization, ease of influence

The draft is currently in IETF last call (IESG review). The chances to have major modifications of the protocol accepted by the IETF are very slim. For example, making the IETF change the protocol to be more suitable for large groups of thin clients by reducing the number of asymmetric operations seems difficult.

Security of the protocol

The registration phase is as secure as IKE phase 1. Once a user is a member of the group the following holds:

- DoS: protection by cookies or signatures
- Replay: messages include a sequence number which makes replays impossible
- Identity protection: same as for IKE phase 1

Validity for different scenarios

The initial IKE phase 1 and 2 (during registration) makes GDOI impractical for extremely large groups, except possibly for messaging. For reasonably large groups it is suitable for conversational multimedia, messaging and content on demand. Hence GDOI is suitable for simple scenarios as Scenario 3, but is to heavy for Scenario 1 and 2.

Ease of deployment

- GDOI requires a PKI to work,
- There is no need for synchronised clocks,
- It is probably easy to integrate GDOI in the MBMS architecture if a PKI is already in place.

GSAKMP-Light

Group Security Association And Key Management Protocol - Light is a registration and re-keying protocol developed by NSA and Sparta.

Number of round trips

- Registration: 1.5 roundtrips. The following messages are exchanged: Join-request, Key-download and Acknowledge.
- Re-keying: 0.5 roundtrips. The re-keying is done by a Key-download message.

Message sizes

- The Join-request message is approximately 1Kb plus the size of the optional certificate.
- The Key-download message is approximately 1Kb plus the size of the optional certificate.
- The third message in the registration is an acknowledgment from the client to the server of a few hundred bytes

Workload

During the registration phase, the following operations are needed:

- Server side: 2 asymmetric key operations + optionally 1 for checking of certificate + 1 SHA-1/DSA. There could be one asymmetric operation less, since g^x can be pre-computed though.
- Client side: 2 asymmetric key operations + optionally 1 for checking of certificate + 1 SHA-1/DSA. There could be one asymmetric operation less, since g^y can be pre-computed though.

During the re-keying phase the following operations are needed:

- Server side: A few symmetric key operations for key-encryption and 1 SHA-1/DSA operation for signing
- Client side: A few symmetric key operations for key-decryption and 1 SHA-1/DSA for checking signing. Optionally also 1 asymmetric operation extra for checking certificate

Implementation footprint

There are no available implementations of GSAKMP, to our knowledge.

Suitability for different AA frameworks

GSAKMP can not make use of any authentication or authorisation performed prior to its execution, but performs all authentication tasks on its own. In other words, it is not suitable to be used with any AA-framework.

Applicability to different security protocols

There is no direct support for any protocol. Though with pre-set algorithms and SA:s, where the key is the only thing that needs to be transmitted, IPsec (v4 and v6), TLS, SMIME can be supported.

It is possible to add parameters to support more or less any protocol, but this implies making another RFC going through the IETF standardisation process.

Algorithms used by protocol itself and their extensibility

The following algorithms are used to manage the group. They are clearly not chosen with wireless/thin clients in mind. Additional algorithms can be used, but requires another RFC.

- Key-creation is done for the re-keying SA using DH in Z_p for some 1024-bit prime p.
- 3DES in CBC-mode to protect key transport during re-keying.
- Signatures done with SHA-1 based DSA.

Status in standardization, ease of influence

The draft is in IETF WG last call. The chances to have major modifications of the protocol accepted by the IETF are very slim. For example, making the IETF change the protocol to be more suitable for large groups of thin clients by reducing the number of asymmetric operations seems difficult.

Security of the protocol

- GSAKMP has no DoS protection,
- Replay protection is implemented by the use of message sequence numbers,
- There is no identity protection.

Validity for different scenarios

Not being able to support pre-shared keys for registration makes the protocol unnecessarily heavy when the operator-subscriber relation exists. Furthermore the signature during re-keying might be too much for large groups/thin clients. Hence GSAKMP is suitable for simple scenarios as Scenario 3, but is to heavy for Scenario 1 and 2.

Ease of deployment

- GSAKMP requires a PKI to work,
- There is no need for synchronised clocks,
- It is probably easy to implement once a PKI is in place, but it is likely to be to heavy on both servers and clients.
- Adding a pre-shared key method to the protocol is possible, but needs standardisation work.

MIKEY

We only consider registration with pre-shared keys, since it is available and the most efficient. It is also suitable for the scenarios.

Number of round trips

- Registration: 1 roundtrip (or 0.5, if no acknowledge is required).
- No re-keying available (though MIKEY could be extended with a 0.5 round trip re-keying). Today a "re-registration" has to be done.

Message sizes

"Push message" of a few hundred bytes and verification message is even less.

Workload

- During the registration phase 1 hash and a few symmetric block encryptions/decryptions for client and server are needed.
- Re-keying is not available (in case of an extension, symmetric keys could be used which would result in a symmetric decryption and a verification of a MAC).

Implementation footprint

A partial implementation (not supporting the DH-exchange) done by Ericsson contains approximately 5600 lines of code (excluding the cryptographic algorithms, which are taken from the OpenSSL library).

Suitability for different AA frameworks

The mode where MIKEY uses pre-shared keys (which is the most appealing one in this case), makes it dependent on out of band signaling, i.e. it must be preceded by a AA-protocol that also results in a common shared key. Hence it is suitable only for USIM/AuC.

Applicability to different security protocols

- MIKEY only supports SRTP today, but it is easily extendable to IPsec etc. Extension requires another RFC though.

Algorithms used by the protocol

- Encryption algorithms: AES in Counter mode
- Hashes: HMAC/SHA-1

Status in standardization, ease of influence

The protocol is in IETF WG-last call and is under Ericsson control, i.e. driven by Ericsson. It is probably easy to extend MIKEY with LKH by a companion RFC and getting it through the IETF.

Security of the protocol

- No identity protection (unless temporary id's are in use),
- DoS protection: responder doesn't do anything before checking the MAC on the message,
- Replay protection done with timestamps (counters could be used if the underlying system would allow to).

Validity for different scenarios

MIKEY is the simplest and fastest protocol for the case where there is an operator-client relation. It does not scale to large groups if re-keying is needed, but this can be remedied by adding an RFC on how to use LKH with MIKEY.

Ease of deployment

- MIKEY does not need a PKI when the shared key variant is used,
- There is a need for synchronised clocks, but the timestamps can be replaced with counters if e.g., AA is used to provide a fresh key for MIKEY for each new session or some other mechanism is used which provides replay protection.
- It is easy to integrate MIKEY in the MBMS-architecture (it can also be piggybacked on SIP/RTSP).

Table of Summary

This section contains a table that summarises the previous sections. It should be noted that the numbers are more to indicate the order of the sizes, required operations etc. That is, they should not be considered as fixed, since all the examined protocols have some amount of flexibility.

Criteria	**GDOI**	**GSAKMP-Light**	**MIKEY**
# of roundtrips	Registration: 5 or 3.5 Re-keying: 2 or 0.5	Registration: 1.5 Re-keying: 0.5	Registration: 1 or 0.5 Re-keying: NA
Msg sizes	Registration: ~1.5kB to ~2.5kB Re-keying: ~1kB	Registration: ~2kB [+ ~3kB] Re-keying: ~1.5kB	Registration: ~500B Re-keying: NA
Workload (asymm. operations)	Registration: 2-3 Re-keying: 1	Registration: 2 [+ 2] Re-keying: 1 [+1]	Registration: 0 Re-keying: NA
Impl. footprint	~35000 lines of code	NA	~5600 lines of code
Suit. for AA	USIM/AUC: yes MBMS: yes Certificates: no	USIM/AUC: no MBMS: yes Certificates: no	USIM/AUC: yes MBMS: yes Certificates: no
Applic. for sec. protocols	Ipsec, TLS, SMIME	None	SRTP
Algos used	3DES & AES in CBC, SHA-1, MD5, RSA, DSS, ECDSS	3DES in CBC SHA-1, MD5, HMAC, DH , DSS	AES in CM, SHA-1, HMAC
Status in std.isation	IESG last call	IETF WG last call	IETF WG last call
Sec. of proto.	DoS: cookies or signs. Replay: SEQ-no Identity: IKE phase 1	DoS: none Replay: SEQ-no Identity: none	DoS: shared key MAC Replay: timestamp Identity: none
Valid. for scenarios	3	3	3
Ease of deploy.	Hard	Hard	Easy

Conclusions

Both GDOI and GSAKMP make heavy use of asymmetric cryptography. As noted above, this might be too heavy on the mobile terminals as well as on the key-management servers. The possibility to change this is very limited. Furthermore, the use of asymmetric cryptography requires a public key infrastructure (PKI). Unless a PKI is already in place, the work of implementing such an infrastructure must be taken into account.

There is no real need for asymmetric key cryptography in the scenarios considered here, when there is an operator-subscriber relationship already established. Hence the most efficient protocol is MIKEY using pre-shared keys, which

only relies on symmetric key cryptography. MIKEY needs to be extended with some sort of multicast re-keying though, i.e. an LKH-functionality. To do this in a standardised way, another draft specifying this must be put through the IETF.

References

[0] Baugher et. al., draft-ietf-msec-gdoi-05.txt, January 2002

[1] Harney et. al, draft-ietf-msec-gsakmp-light-sec-01.txt, July 2002

[2] Arkko et. al. draft-ietf-msec-mikey-04.txt, September 2002