Security Measures in IP-Multicasting – Classification and Comparison

Christian Grosch
FernUniversität Hagen
Fachgebiet Kommunikationssysteme/FTK
Feithstr. 142, D-58084 Hagen, Germany
Email: christian.grosch@fernuni-hagen.de

Abstract.
The attractiveness of applications for multipoint communication increases with the expanding availability of conference partners in the Internet. Especially in the context of group communication for business applications the demand for security services becomes more and more important, because the information exchanged is usually much more sensitive than in private context. Therefore the demand for data and conference member authentication, privacy, and data integrity increases. It could be seen that, for different reasons, the new developments of multicasting techniques are mostly not considering security measures in an adequate way.

This paper first summarizes the different kinds of applications which use multicasting or are expected to do so in future. They are classified by characteristics like group size, membership dynamic, and need for security services. Then a general survey of the most important proposals for providing security services and key management is given and the protocols are analyzed, focussing on the load for the group controller as the central key managing entity.

The paper closes with a new proposal for protocol improvement for the general process of rekeying.

1 Introduction and Motivation

During recent years the tremendous growth of the internet community has lead to a number of problems in the IP environment. The underlying protocols were intentionally not designed for the millions of users it supports nowadays. A wide variety of difficulties results not only from the growing number of participating hosts, but also from a changing spectrum of applications, data types, and security measures.

In the area of group communication, the fact that potential conference partners are present in the networks to a growing extend leads to an increasing attractiveness and use of new forms of communication, based on relationships like one-to-many, many-to-many or some-to-many. These forms of communication are supported by various multicasting techniques integrated in the IP family. The prototype of a multicast backbone, the MBone project, once instantiated as an experimental overlay of the Internet for limited time and purpose gains increasing attractiveness. Its technology is often considered as fundamental for new kinds of applications and services.

The use of the public Internet has some important disadvantages. As a packet switched network, information is directly available in the network and can be misused both in the unicast and the multicast environment. Appropriate countermeasures have to be installed. This paper presents several security services in multicast environments and is structured in the following sections.

Section 2 introduces applications and service types actually seen as the most common ones by various experts. The typical scenarios and network and security requirements are listed and a short classification is given.

Section 3 introduces different proposals currently available, which deal with the problems of providing security services in multicast environments. Although they all vary in some aspect or another they are mainly based on a limited number of different concepts of key management which are presented.

In section 4 different concepts are compared with each other. The comparison focuses on the actions the group controller has to perform as the central key management entity. The formal analysis is further verified by the outcome of various simulations.

Based on the results presented in section 4, a modification of the general process of rekeying is proposed in section 5 to improve the performance behavior for groups of large sizes and/or high dynamic.

2 Multicast Applications and Security

New types of applications are evolving driven by the increasing availability of network capacity, potential customers, and conference partners. These applications can be distinguished by means of the characteristics type of multicast (one-to-many/many-to-many), number of group members in one session, number of parallel sessions of the same type, group dynamics and topological distribution of session members.

As the kind of applications and the relationship between the involved group members is very different in respect to multicast characteristics, there is also a wide variety of different objectives concerning security services. The most obvious ones are confidentiality and sender and data authenticity. More topics have to be considered: access control, non-repudiation, announcement security, anonymity, and copyright protection. Table 1 gives an overview over the relationship between different applications, their multicast characteristics, and their security requirements. The purpose of the table is to give an impression of the spectrum of possible combinations.

Table 1: Application types, their security services, and multicast characteristics

	bco	ptv	cscw	btv	tedu	dsim	ddb	info	gam	mbo
confidentiality	x	x	x	x	x	x	x	x		x
authentication	x	x	x	x	x	x	x	x	x	x
access control		x	x	x	x		x	x	x	
anonymity		x						x		x
non-repudiability	x		x				x	x		
announcement security	x									
one2many/many2many	m	o	m	o	m	m	o	o	m	m
group dynamic (high/low)	l	h	l	l	l	l	l	h	h/l	h/l
distribution (sparse/dense)	s/d	s/d	s/d	s/d	s/d	d	s/d	s/d	s/d	s/d
group size (huge/small)	s	h	s	h	s	h	s	h	s	s
traffic (high/low)	h	h	l	h	h	h	h	l	h	h

bco-business conferencing, ptv-pay tv, cscw-computer supported collaborative work, btv-business tv, tedu-teleeducation, dsim-distributed simulation, ddb-distributed databases, info-infomation channels, gam-gaming, mbo-mbone

The table illustrates where the main focus in research work lies in. The secure authentication of conference partners and the confidentiality of transmitted data are the most important services, because they apply to nearly every group communication scenario. As the combination of all mentioned services in the different applications is very difficult, a compromise must be found between feasibility, performance aspects, and demand for security services.

3 Related Work

For achieving some or all of the security objectives mentioned in section 2, a number of proposals is available. The following list gives a short survey on the most important ones and references for further reading.

The Group Key Management Protocol (GKMP) [5] presents a scheme with one central entity called group controller (GC), which is responsible for performing authentication, access control, group key generation, and its secure distribution to all registered members. For the process of key distribution, the GC contacts each member individually and proves its identity. Then they establish a secure channel by Diffie-Hellmann based generation of a session key and exchange all data necessary for performing the join operation. This process consists of three states, whereby the consultation of a trusted third party like a trust center for signature verification is not included. As in the proposal of the GKMP, this paper does not deal with trust center functions and communications. The presence of a public-key infrastructure is presumed given for reasons of simplicity. GKMP stands for protocols, which deal with each group member directly for each management action necessary.

The Framework for Group Key Management for Multicast Security (FGKM) [4] is an example for a hierarchical group key management concept. The network is subdivided into several distinct areas, called the *leaf* and the *trunc* regions. All group senders and receivers are situated in leaf regions. There is no group member directly connected to the trunc region, whose purpose is to multicast the traffic from the sender's region to each region containing active receivers. Each region uses its own multicast and security protocols and session keys. Join and leave requests affect therefore only the local domain and its limited number of group members. The disadvantage of this concept is the need for traffic decryption/encryption at the border of each leaf region to the trunc region. Another proposal based on hierarchical structures and subdivisioning of networks in distinct areas is Iolus [7].

The Key Management for Multicast (KMM) proposal from Wallner, et. al. [8] presents an efficient solution for support of dynamic group membership. Similar to GKMP, hosts wishing to receive group data send their join messages directly to the designated group controller which is responsible for secure generation and distribution of group keys. For an efficient key management the keys corresponding to the members are organized in a binary tree structure. Using an intelligent algorithm for the rekey process, the complexity of this task could be reduced to $O(log_2(N_h))$ with N_h referring to the number of group members.

The analysis presented in this paper is related to this three basic concepts. Further proposals and surveys to this special area of research can be found in [1, 3].

4 Analysis and Comparison

Looking for performance estimations, in the papers mentioned above only general statements in terms of complexity could be found, in most cases by means of network load for security protocol overhead and memory usage for storage of keys in all included entities: senders, receivers, and group controllers[1].

This paper focuses on tasks the group controller as central group manager has to fulfill and especially analyses how heavy the burden of performing up to several hundreds or thousands of cryptographic calculations per second is. Several protocols avoid to use a fixed group controller entity for more than one group and/or session by introducing

[1]The term group controller (GC) is used representatively for the corresponding terms in the various papers and proposals. Depending on the related protocol, it also stands for several distributed entities.

Table 2: Results for the Crypto++ Benchmark on a Pentium II 300MHz

Operation	Time/Operation	Operation	Time/Operation
RSA 1024 Encryption	$t_e = 1ms$	RSA 1024 Decryption	$t_d = 46ms$
RSA 1024 Signature	$t_s = 46ms$	RSA 1024 Verification	$t_v = 1ms$

the concept of choosing a receiving host for that purpose by random or predefined rule. Considering this concept, it is clear that the group controller process runs on a machine of average performance and equipment. This leads to the question of how many tasks such a machine is able to perform without blocking the whole system.

The analysis performed follows a straight forward strategy. The three protocols GKMP, FGKM, and KMM are examined by counting the exchanged messages in the different stages of the algorithms and the desired cryptographic calculations at the GC.

4.1 Cryptographic Calculations

All examined protocols and proposals are based on standard cryptographic procedures and algorithms. The basic cryptographic actions are: key generation (stand alone or joint), data encryption and decryption (asymmetric and symmetric algorithms), signature and verification of signed data.

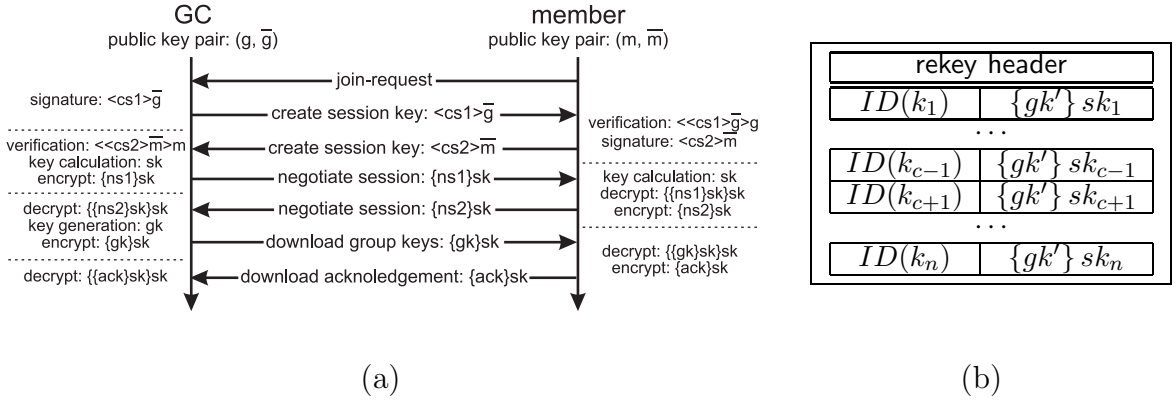
For an estimation of the load these calculations are implying on the group controller and the receiving hosts, the benchmark results of the Crypto++ Library [2] have been used. The benchmark was run on a Pentium II 300MHz machine under Windows 98. The sources were written in C++ and the code was generated with compiler code optimization for speed. This machine can be seen as an equivalent for an average host machine joining the group and the measured times are suitable for a first rough estimation. Table 2 presents the values used in this analysis. It is assumed, that the significantly faster operations of encryption and verification are resulting from the choice of a short and optimal key as proposed e. g. in [6].

4.2 Protocol Simulation

The theoretical results from the analysis allow a first estimation of the overall performance, but a network environment represents a complex system for which it is difficult to consider all influencing elements. The simulation has been implemented mainly for two purposes: (a) to measure the overall delay time of the processes of initialization and rekeying and (b) to underpin the assumption that the network load overhead implied by the security protocol is small compared to the data traffic produced e. g. during video conferences or pay-per-view sessions.

The Opnet simulation tool was chosen as simulation environment and a simplified model of the German MBone (`http://www.mbone.de/`) has been chosen, and modeled as basic network topology. The resulting simulation network consists of a hierarchy of $N_d = 63$ subnets connected by the backbone. The subnets themselves are simple and consist of one router and $N_s = 10$ hosts each, so for the simulations there is a maximum of $N_h = 630$ hosts.

For the initialization process the parameters for the simulation were the number of hosts going active at least once during a simulation run and the span of time during which join requests occur. Simulating the process of rekeying, one single operation has been evaluated for each run, again varying the group size.



(a) (b)

Figure 1: Group Key Management Protocol: join (a) and rekey (b)

4.3 Session Initialization

It is obvious, that the load for the GC increases, when the amount of requests per time rises. Therefore the analysis and especially the simulation focuses on the processes of initialization and rekeying, in which the maximum portion of group members are involved in.

Performing the first step of building up a communication relationship, two host have to pass a predefined number of states: (1) authentication, (2) secure channel establishment, (3) distribution of group or session keys, and optional acknowledgement. Figure 1 illustrates the principle. Regardless of the number of messages being exchanged between the hosts, the group controller has to calculate and verify a signature, he has to calculate his part in the key generation process (e. g. Diffie-Hellman based protocols), and he finally has to distribute the encrypted group specific information containing the group keys. For the GKMP and the KMM concept figure 1 (a) illustrates the chain of operations.

Summarizing the corresponding times from table 2, the overall time needed for the execution of the necessary algorithms for the join process leads to $t_j = t_s + t_v = 47ms$. This means, that the maximum number of join-requests per time, which could be handled by the GC could be calculated as $n_j = \frac{1}{t_j} = 21,3s^{-1}$. The operations for symmetric key operations are left out in the equation, because they are at least by factor 10^2 faster and therefore unimportant for the time estimation.

Figure 2 (b) shows the simulation results for the join process. The group controller has a maximum throughput, which is slightly lower than the calculated optimum. Reduced by the overhead for key storage and general system and network management, the GC performs an average of $\bar{n}_{j,sim} = 16.8s^{-1}$ join processes. After exceeding this average value, the mean delay time for the join process grows linearly with the number of group members. The average minimum time for joining a secure session is $\bar{t}_{j,min} = 220ms$. This time varies for the individual host, depending on the distance from the GC and the number of routers the messages have to pass.

The FGKM protocol performs significiantly better for the join process. Since the local GCs are responsible only for the N_s hosts of their own domains and the backbone key has to be distributed by the global controller entity or intermediary to each subnetwork only once, the heavier load results for the entity, which has to service the most clients. For the simulation example, the performance improvement could be weighted by factor $N_{fgkm,max} = max(N_d, N_s)$. Figure 2 (b) illustrates the simulation results for this case.

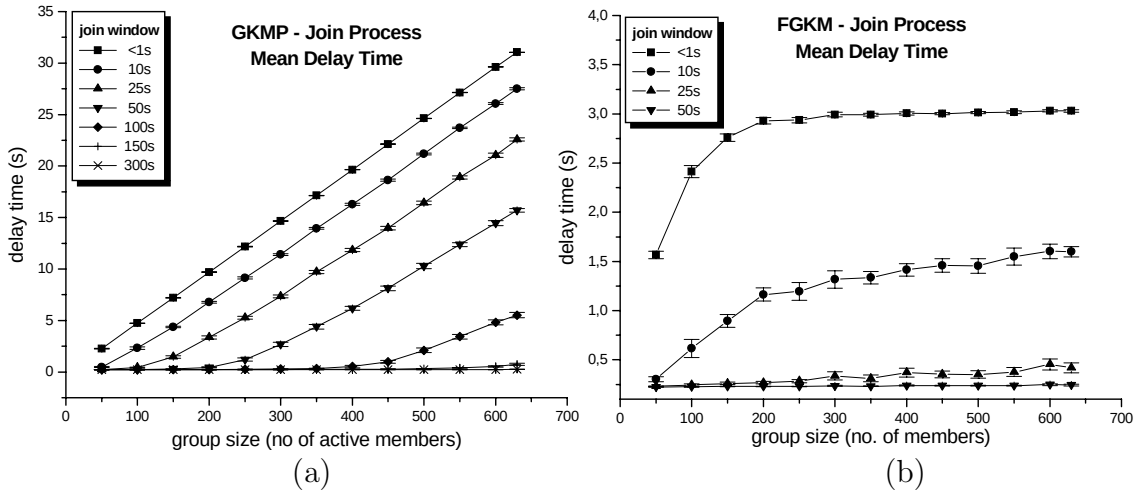


Figure 2: Simulation results for the join processes of GKMP (a) and FGKM (b)

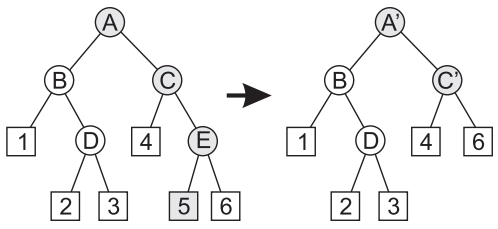
4.4 Process of Rekeying

If the group is compromised a new group key has to be established. As mentioned before, the storage of the individual session keys is encouraged because it simplifies the renewed establishment of a secure channel and therefore significantly speeds up the operation.

For the GKMP, the process of rekeying could be executed in two ways. It is possible to encrypt the new key with the corresponding session key of each individual host and transmit this message by unicast. Alternatively all encrypted keys could be grouped together and be send out by multicast. For allowing the hosts to identify the correct key, some ID-token has to be added. Figure 1 (b) illustrates the new key packet for the case, that host C compromizes the group.

Figure 3 (a) shows an example for the KMM protocol, using a binary tree for efficient key management. Since host 5 leaves the group, all keys he possesses are regarded compromised and must be renewed. In analogy to GKMP, this could be done by sending the packet of figure 3 (b) to all host via multicast. The efficiency of the KMM protocol depends on the layout of the tree. The number of new keys to be generated and corresponding messages to be sent depends on the average depth of the binary tree. Figure 3c shows the average depths measured from the simulation runs. The outcome of the simulations is comparable with the outcome of a stochastic process for this case.

For the FGKM the problem of rekeying only applies to the subgroup where the compromising host resides in. Therefore the process is identical to the one executed for the GKMP, only altering the number of affected host to N_h.



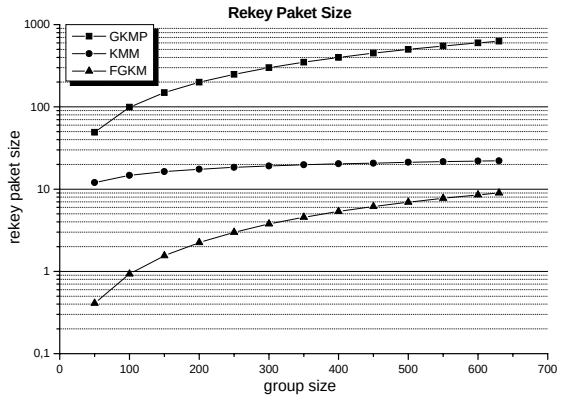
rekey header	
$ID(6)$	$\{C'\}\,sk_6$
$ID(4)$	$\{C'\}\,sk_4$
$ID(B)$	$\{A'\}\,B$
$ID(C')$	$\{A'\}\,C'$

(b)

GS	mean	max
100	9.4	14.5
200	10.5	16.6
300	11.8	18.4
400	12.1	19.3
500	12.7	19.9
600	13.0	20.6

(c)

Figure 3: Rekey Process of KMM



	Initialisation	Rekey
GKMP	$O(N_h)$	$O(N_h)$
FGKM	$O(max(N_d, N_s))$	$O(max(N_d, N_s))$
KMM	$O(N_h)$	$O(log_2(N_h))$

Figure 4: Comparison for the Rekeying Process

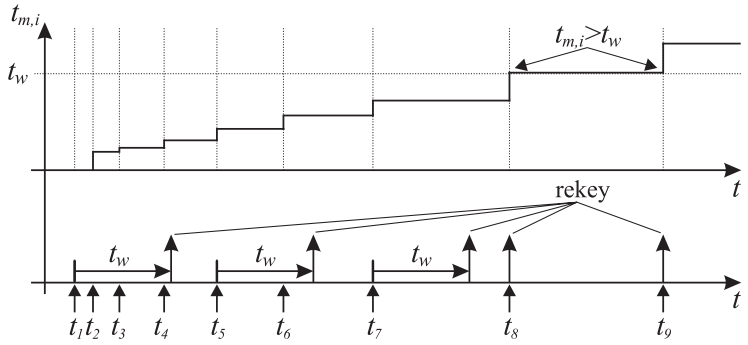
Figure 4 presents an comparison of the operation complexity for the different protocols.

5 General Improvement of the Process of Rekeying

Regarding the different proposals and surveys available, a group key is normally regarded compromised when a host leaves a group. The leaving member possibly made a copy of the actual key before sending the message and could now use this key to decrypt the multicast packets available without being registered by the group controller.

As the results from section 4.4 demonstrated, the process of rekey could be very time and resource consuming, especially if the groups are huge and/or show a very dynamic behavior. To reduce the protocol overhead for the security services, a *minimum wait time* t_w is introduced. On receiving a leave message from one of the registered hosts, the GC starts a rekey timer with initialization time t_w. While he waits for the timer to reach zero, he receives and collects further leave messages from other hosts. Finally the rekey process is started and all queued messages are processed at once. Figure 5 illustrates the idea.

This concept can be applied to several forms of application, for which it is acceptable to offer some potential extra time of service for free in exchange for a significant reduction of protocol overload and increase of GC performance. If we assume that a session has an average leave rate of n_l, e. g. $n_l = 10s^{-1}$, the average number of leave



i	t_i	$t_{m,i}$	t_r	t'_r
1	5.14	0.00	5.00	5.00
2	6.09	0.88	4.05	4.05
3	7.46	2.14	2.68	2.68
4	9.78	4.27	0.36	0.36
5	12.51	5.91	5.00	5.00
6	15.96	7.83	1.55	1.55
7	20.60	9.96	5.00	5.00
8	27.68	13.96	5.00	0.00
9	35.64	18.11	5.00	0.00
mean delay			3.74	2.63

(a) (b)

Figure 5: Illustration of the waittime concept

messages, being collected during the wait interval is $n_w = n_l * t_w$. If we assume further a wait time of $t_w = 5s$ in the average case, the number of rekey processes is reduced by factor 50.

The concept could be improved further, if the interarrival time for leave messages is monitored. If the average interarrival time

$$t_{m,i} = \frac{1}{N_w} \sum_{j=0}^{N_w-1} t_{i-j} - t_{i-j-1} \tag{1}$$

for the last N_w arrivals becomes larger than t_w, the rekey process is started at once. Considering this modification, the average time span $\bar{t_r}$ for a leave message in the queue to be served could be reduced. In the example in figure 5 N_w was set to 3 and the modification reduces the wait time by 30%.

6 Conclusion

In this paper, the results of a comparison of the most important protocols for multicast security were presented. The comparison concentrates on the impact of cryptological algorithms on the performance of the group controller as the central management entity of most proposals. By analyzing the basis tasks for group initialization and rekeying after key compromisation, a rough estimation for characteristics like process times, throughput rates, and maximum group sizes are given. The outcome of various simulations based on the topology of the German MBone and implemented using the Opnet simulation tool was used to confirm the theoretical estimations.

Acknowledgement

The research work presented in this paper has been done at the Department of Communication Systems/FTK of the FernUniversität Hagen. The author thanks Prof. Dr.-Ing. Firoz Kaderali for the supervision of his work.

References

[1] Anthony Ballardie. Scalable multicast key distribution. *Internet RFC 1949*, 5 1996.

[2] Wei Dai. Speed comparison of popular crypto algorithms - crypto++ 3.0 benchmark. *Internet: http://www.eskimo.com/ weidai/benchmarks.html*, 1 1998.

[3] Li Gong and Nachum Shacham. Elements of trusted multicasting. In *IEEE Proceedings of International Conference On Network Protocols, Boston*, 10 1994.

[4] Thomas Hardjono, Brad Cain, and Naganand Doraswamy. A framework for group key management for multicast security. *Internet-Draft: draft-ietf-ipsec-gkmframework-00.txt*, 7 1998.

[5] Hugh Harney and Carl Muckenhirn. Group key management (gkmp) architecture. *Internet RFC 2094*, 7 1997.

[6] Alfred Menezes, Paul van Oorschot, and Scott Vanstone. *Handbook of applied cryptography*. CRC Press, 1997.

[7] Suvo Mittra. Iolus: A framework for scalable secure multicast communication. In *Proceedings of the ACM SIGCOMM '97, Cannes*, 9 1997.

[8] Debby M. Wallner, Eric J. Harder, and Ryan C. Agee. Key management for multicast: Issues and architectures. *Internet-Draft: draft-wallner-key-arch-01.txt*, 9 1998.