A Key Distribution Architecture for Encrypting Multicast Communications

M. St Amand [1], B. Amar Bensaber [1], I. Biskri [1], A. Zebdi [1], D. Seret [2]
[1] *Department of Mathematics and Computer Science*
University of Quebec at Trois-Rivières, Canada
3351 Bd des Forges C.P 500 - Trois Rivières, G9A 5H7 - Québec – Canada.
Email : {Bensaber, Stamandm, Biskri, Zebdia }@uqtr.ca
[2] *UFR de mathématiques et informatique*
Université René Descartes, Paris 5 - 45 rue des Saints Pères, 75270 Paris cedex 06
Email : Dominique.seret@math-info.univ-paris5.fr

Abstract – In this work, we propose architecture for key management in multicast communications. This architecture combines the complementary variable approach with slided keys and hierarchical binary trees. Using this architecture, it will be possible to reduce the number of transmissions as one or more participants join or leave a group. Also, it includes a technique for the name attribution of keys, which allows to increase the clearness of the hierarchical binary key trees and to facilitate the treatment when refreshing keys. Many results of the simulations done using C++ code and Opnet Modeler are presented.
Keywords: Secure Multicast, Key Distribution, Hierarchical Binary Trees, Slided Keys.

1 Introduction

In Multicast communications, an entity (transmitter) sends information packets to receivers (participants). During the transmission over insecure network like Internet, the packets conveying the data can be easily read by third-parties. If the data need to be secured, a cryptography method must be used to guarantee data confidentiality.
An entity called the Key Distribution Center (KDC) is attached to the transmitter and is responsible to create keys that will be used to encrypt the data sent to the group. The method of encrypted transmission must preserve the group secrecy so that an entity that does not belong to the group and willing to join the group has access to the information transmitted within the group only after her join and before her leave. It must also preserve the backward secrecy so that a participant having left the group has not access to the information transmitted after its leave. Finally, it must preserve the forward secrecy so that a participant joining the group has no access to the information transmitted before its arrival.
The methods used for encrypting point-to-point communications cannot be used in Multicast communications, because they consume considerable network resources when performing key distribution [1]. Few encryption methods applicable to Multicast communication have been proposed. An optimization of these methods must be carried out to reduce the time for keys' sending and the encryption computation time. A rigorous study of the operations needed when a participant joins and leaves would make it possible to find a better way of performing these operations.
The method that we propose is based on combining the complementary variable approach [1], with slided keys and hierarchical binary trees [1,2]. This method will make it possible to reduce the number of necessary transmissions while preventing the collusion between several participants.
Also, this method includes a technique for naming the keys in the tree which allows to increase the clearness of the hierarchical binary key trees and to facilitate the treatment when refreshing keys. This new method can also be used to find which participant of the group knows a certain key.
Our paper is organized as follows: first of all, we present a review on the hierarchical management of the keys; second, we propose our solution (RS or ROKeN-SKI) with all the necessary architecture to its modeling. Then, we describe the technique of keys' refreshing which we use and we define the various

identifiers and keys necessary to the encoding of the transmitted data. Thus, we detail the group management operations and the finite state machines of each entity. Finally, we present the results of the simulations.

2 Related work

The contribution of the Hierarchical Tree Methods (HBT) [1,2] for the transmission of the keys made it possible to reduce the transmissions of keys and opened new possibilities towards optimization. Below are some interesting methods derived from HBT.
The goal of the method proposed by A. Perrig [4] was to reduce the amount of bandwidth used during the transmission of the keys, by reducing the size of the packets. The ELK [4] method uses mathematical methods and *indexes* to reduce the size of the packets when sending keys. When the participant entity receives the packet and the indexes at the same time, it could generate the keys with indexes. Indexes are thus a generator of remote keys. Perrig seems to be the only one to be concerned with losses of keys. He proposes that the indexes can be sent in the same packets as the data. This proposal is interesting since the data are resent by the methods with a reliable Multicast retransmission. In this case, the operations of keys' loss of would be nonexistent and the cases where a participant would not be able to decipher a packet are reduced to zero.
In approaches like OFT [5] and EHBT [3], *hidden keys* are used. This technique is very interesting due to its capacity to regenerate keys with information held by the participants. In the "One-Way Function tree" method, OFT [5], each key has its hidden key that is used to regenerate a new key. When adding a participant, the KDC sends the hidden keys and the keys the participant may need in a Unicast packet which can consume a lot of bandwidth. This technique uses the keys of left son or right son to regenerate the keys. Furthermore, the keys can be refreshed by the utilisation of the indexes or derivation of a key of its sons and be computed in a one-way function. The Enhanced Hierarchical Binary Trees technique, EHBT [3], also uses a hidden key for each key and it is stored in the brother of the key's owner. The KDC sends the higher keys encrypted with this new key in the same way as in the hierarchical trees. The technique also uses *command sets*, hidden in the packets, which allow the participant to know the orders to be carried out. The packets also contain indexes indicating necessary structures to know which key to use to regenerate the new keys. In the taxonomy suggested by Canetti et al. in [6,7], the authors propose to use the lower or higher part of a key to regenerate another, but it is more difficult for an external entity to find the value of the key. It uses the high or low part of the key's left or right son to regenerate a key. It also uses Traffic Encryption Keys and Key Encryption Keys with numbers in packets to save transmissions during keys' refreshing.

3 The ROKeN-SKI method

The goal of our research is to find which operations should be done to decrease as much as possible the transmissions between the KDC and the participants in the keys' distribution.
Our method, ROKeN-SKI or RS in this paper (which stands for Refresh-Oriented Key Naming and mixed Slided Keys and Identifiers method), proposes an architecture with a variable level of identifiers (also called complementary variables [1,2]) mixed with slided keys (also called hash keys). The levels start from the bottom of the tree and up to the root. The number of levels is correlated with the number of participants knowing the identifier associated with a participant.
The KDC chooses a level of identifiers to be used. The chosen level sets the needs for packets to be transferred when a participant leaves. Identifiers will be sent at the creation of the group and upon joins. In the leaving of a participant, a high level of identifiers will generate small amount of packets at the leave of a participant. A low level will create a high amount of packets. A very low level of identifiers will be like the HBT [1,2] approach.
A more complete description of the RS method is available in the references [14] and [15].

3.1 Names of keys

The name of the group key (the root) is "m". The following keys are named according to their position in the binary tree. The name of a key is his parent name and its added number. If the key is the left son of -its father, it receives a zero (0) at the end of its name but if it is the right son, it receives a one (1) at the end of its name. So, the name of the right son of the root is 1 and the left son is 0. The grandsons of the root will carry the names 00, 01, 10 and 11.
The name of a particular key provides the names of its parents, from its up to the root. The maximum length of the key name is the height of the tree. This feature will be used in the keys' refresh method.
The name of a participant is given by the name of its personal key in the keys' tree.

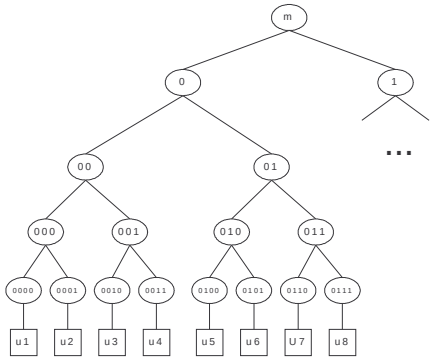


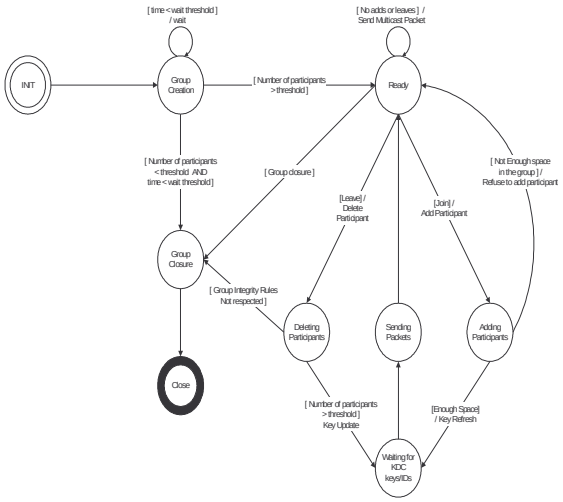
Figure 1: Names of keys

3.2 Keys' Refresh

On a join of a participant, each participant must refresh the keys that the new participant will know, to preserve the backward confidentiality of the Multicast communication. The name of the participant, given by its place in the tree, determines which keys will be transparent to it. Thus for each join of a participant, the KDC must refresh all the keys connecting the position of the new participant to the root of the tree [14,15].
To indicate which keys needs to be refreshed, the KDC sends a Multicast packet *REFRESH*. In the packet REFRESH, the *NAME* field contains the name of the key that created the need for a refresh: the new participant's personal key. The informations contained in its name are very useful: they contain the names of all the keys requiring a refresh. By this way, a participant will know which key needs to be refreshed, without needing a structure of orders in the packets and will not need to refresh all its keys: only those it has in common with the new participant.
When a participant receives a REFRESH packet, with a name of a key in field NAME, it refreshes the group key and then checks the family of the keys to be refreshed: all the parents of this key need to be refreshed. It refreshes all the keys it has in common with the new participant.

3.3 Operations on the group

Here are the finite state machines of each entity. For the states JOIN and LEAVE, the number of states is equals. Details of the operations are provided below.



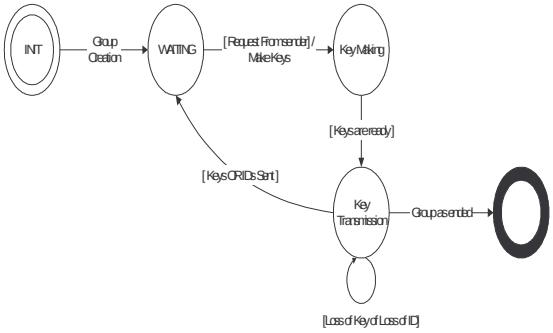


Figure 2: The KDC entity's Final State Machine

Figure 3: Sender entity's Final state machine

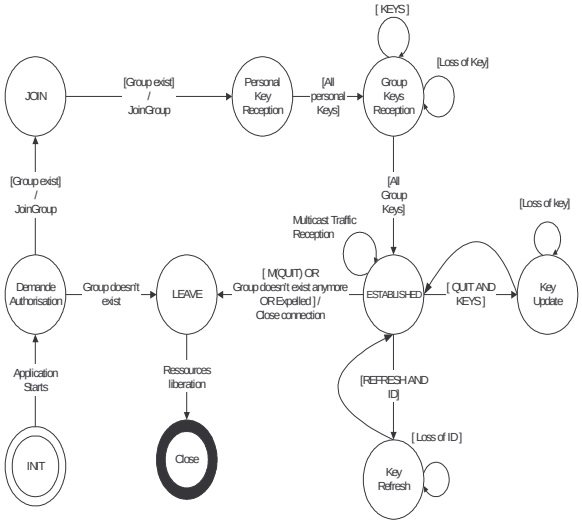


Figure 4: Participant entity's Final State Machine

3.3.1 Simple join

When a new participant joins the group (simple join) the KDC makes the new keys intended for this new participant and pulls the participants already in the group to refresh the keys that will be used by the new participant. A Multicast packet REFRESH will be sent to inform the group about the name of the new participant and the keys it intends to use. The KDC then sends to the participants who will be cousins of the new participant the identifiers associated with this participant (*COUSIN* packet). Finally, it sends the keys to the new participant (*KEYS* packet), as well as the identifiers of its neighbours (*COUSING* packet).

When a simple joins occurs, the participants receive a REFRESH packet and must carry out the refresh functions. If they have keys shared with the new participant they will have to make a refresh operation

and if they are one of the cousins of the new participant they will have to carry out a treatment on each level of identifiers. The new participant receives the keys allowing it to decode the future packets (KEYS packet) and the identifiers of its cousins (COUSING packet). It stores information in the identifiers table and in the keys' table.

3.3.2 Multiple Join

The main differences in a multiple join are that a *REFRESHG* Multicast packet is used, with all the names of keys needed to be refreshed. The operations remain the same as in the simple case.
When a multiple join occurs, the KDC makes the new keys intended for these new participants and tell the participants already in the group to refresh the keys that will be used by the new participants. A Multicast packet REFRESHG will be sent to inform the group about the name of the new participants and the keys intended to be used. It then sends to the participants who will be cousins of the new participants the identifiers associated with those participants (COUSING packet). It finishes by sending the keys to the new participants (KEYS packet), as well as the identifiers of their neighbours (COUSING packet).
When a multiple joins occurs, the participants receive a REFRESHG packet and must carry out the refresh functions. If they have keys shared with the new participants they will have to make a refresh operation on them and if they are one of the cousins of the new participants, they will have to carry out a treatment on each level of identifiers. The new participants receives the keys (KEYS packet) allowing them to decode the future packets and the identifiers of their cousins (COUSING packet). They store information in the identifiers' table and in the keys' table.

3.3.3 Simple leave

When a simple leave occurs, the KDC sends a Multicast *QUIT* packet with the name of the leaving participant in the NAME field. Depending on the level of identifiers used, it will have to transmit all the keys which will not be regenerated with the identifiers technique. Those keys will have to be sent to all the participants who do not know the keys used in the technique of the slided keys to regenerate each key of tree, known by the leaving participant.
The participants who do not leave the group receive a Multicast QUIT packet and do an update of some of their keys: those which are known by the leaving participant. They must check how many levels of keys they must regenerate with the identifiers of their neighbours and how many levels of keys they must regenerate with the slided keys. The keys which are not updated with the identifiers will be regenerated with the slided keys. If they know the keys that will be used to regenerate the keys used with the slided keys then they use the one-way function with the key in parameter, otherwise, they must wait the transmission to receive the new value of the key.

3.3.4 Multiple leave

The main difference in a multiple leave is that a *QUITG* Multicast packet is used, with all the names of the leaving participants in it. The operations remain the same as in the simple case.
When a multiple leaves occur, the KDC sends a Multicast QUITG packet with the names of the leaving participants in the NAME fields. Depending on the level of identifiers used, it will have to transmit all the keys which will not be regenerated with the identifiers technique. Those keys will have to be sent to all the participants who do not know the keys used in the technique of the slided keys to regenerate each key of tree, known by the leaving participants (in a *KEYS* packet).
The participants who do not leave the group receive a Multicast QUITG packet and do an update of some of their keys, those which are known by the leaving participants. They must check how many levels of keys they must regenerate with the identifiers of their neighbours and how many levels of keys they must regenerate with the slided keys. The keys which are not updated with the identifiers will be regenerated with the slided keys. If they know the keys that will be used to regenerate the keys used with the slided keys then they use the one-way function with the key in parameter, otherwise, they must wait the transmission to receive the new value of the key.

4 Results

In this section, we introduce the simulations results. The simulations were done using C++ code and OPNET Modeler 9.1A. These simulations compare our protocol RS to HBT protocol.

4.1 C++ simulations

To be sure that our technique is working well, we first implement it using C++ code (Visual Studio environment). We used groups that vary from 1000 to 30 000 participants. We calculate the number of packets generated during a join and a leave of participant(s).
With these results, it is clear that the RS method:

1. Uses less packets than the HBT method on a join or a leave of a participant;
2. As the number of levels used is increasing, the quantity of packets increases too. This excludes the levels 0 to 4 where there is no growth in the quantity of the packets.

These simulations permit to state that:

1. Between level 2 and 5, the number of packets exchanged on a join and a level are more equalized;
2. The use of the network will be less if the balance between the quantity of packets in a join and the number of packets on a leave is normalized.

As we can see in figure 6, the RS method is scalable. Even with a high number of participants, the method preserves its properties.

4.2 OPNET simulations

For the OPNET simulations, the RS method with 3 levels will be used in a comparative study with the HBT method for the single join and single leave.

4.2.1 The simulation parameters

We keep track of the simulation parameters to have a real comparison between the two methods. The HBT method (in blue) and the RS method (in red) have been simulated with equal simulations details and with the same events.
The four next figures (figures 7, 8, 9 and 10) show these parameters. The x's shows the simulation time (m for minutes) and the y's show the frequency.
The number of participants has been defined by a normal distribution. The two methods have the same amount of participants with a little difference.
The four next figures (figures 7, 8, 9 and 10) show these parameters. The x's shows the simulation time (m for minutes) and the y's show the freqThe number of participants defines the numbers of levels in the tree of keys. The next figure (figure 8) shows the number of levels in the tree of keys. Again, the two methods look similar.
To choose the moment at which each participant joins the group is given by a normal distribution. The same law is used for the time of the participant's session length. The quantity of joins and leaves is shown in the next figures (figure 9, 10). Again, the numbers are almost identical.
With all these figures, we have shown that the two methods have been used with the same parameters.

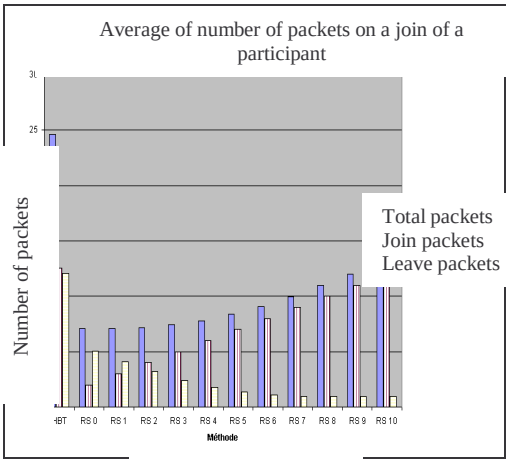
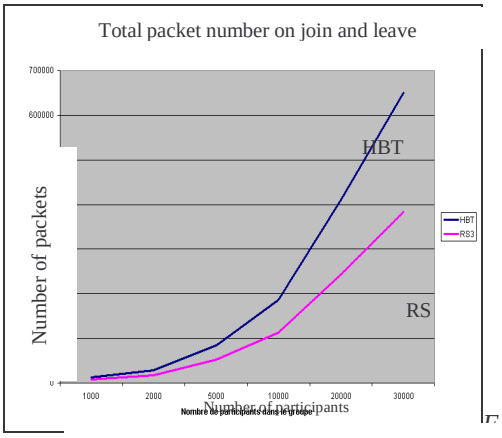


Figure 5: a join or a leave of a participant



igure 6: Number of packets transferred depending on the number of the participants

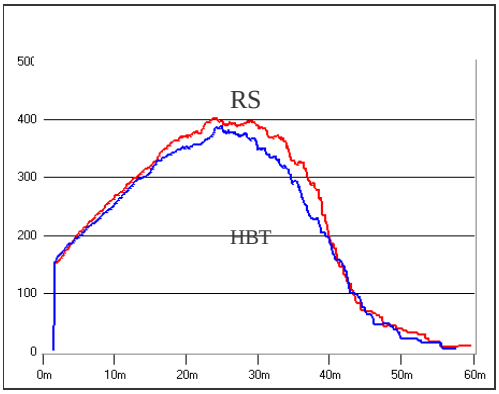


Figure 7: Number of participants depending on the time

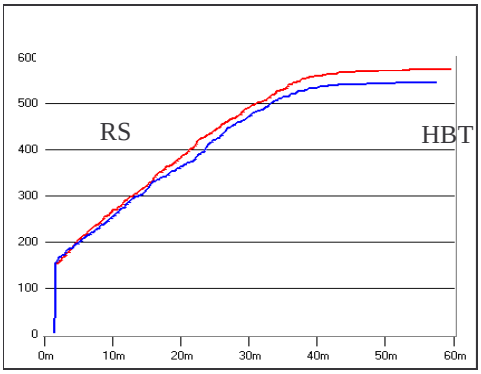


Figure 9: Number of joins

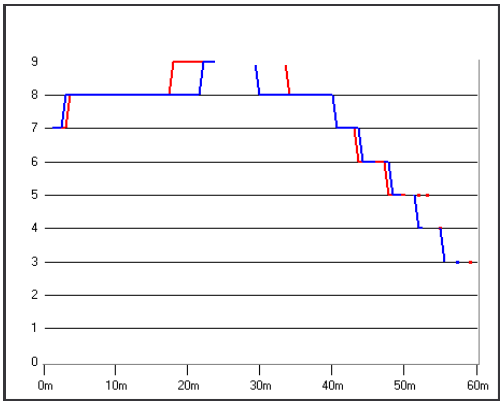


Figure 8: Number of levels of the key's tree

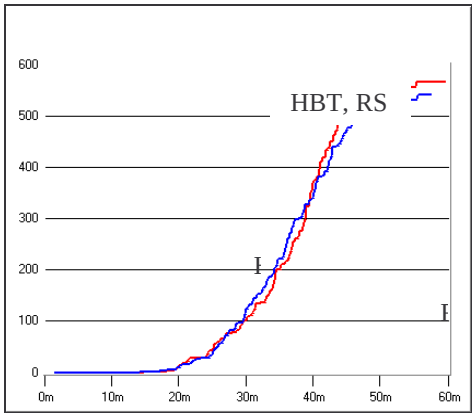


Figure 10: Number of leaves

4.2.2 Comparisons between HBT and RS

4.2.2.1 Introduction

Few parameters exist to compare two Multicast methods. The first is the number of packets exchanged and the second is the computation demands.

4.2.2.2 Packets exchanged

The number of packets exchanged can be split in two types: Multicast and Unicast. Packets have been captured on joins and on leaves. The number of packets exchanged in the regular exchange of information has not been simulated. In the beginning of the simulation, the number of packets exchanged is similar. But, as soon as the number of participants is increasing the differences between the numbers of packets exchanged become higher.

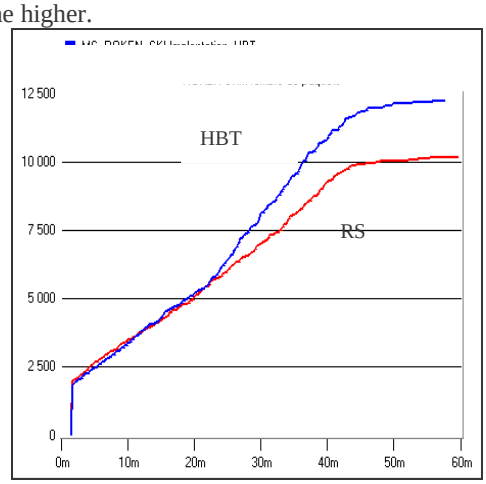


Figure 11: Number of packets exchanged

We could also detail this result with the number of Multicast and Unicast packets. Sending a Unicast message to all participants is expensive compared to a Multicast packet. The goal is to substantially reduce the number of global Unicast packets.
In the simulations, the number of Unicast packets is great with the HBT method. This tends to state that the HBT method is less efficient than the RS method, even when the number of Multicast packets is greater with the RS method.
On a join and a leave of a participant, it is significant to analyze what is going on to understand what is in stake. One side of our method is to balance the number of exchanged packets on the join and on the leave. The next figures will show the reliability of our method.
On a join of a participant, the number of packets is superior with the HBT method. Even when the number of Unicast packets is less, the sum of the Unicast and Multicast packets is greater with the HBT method than with our method.

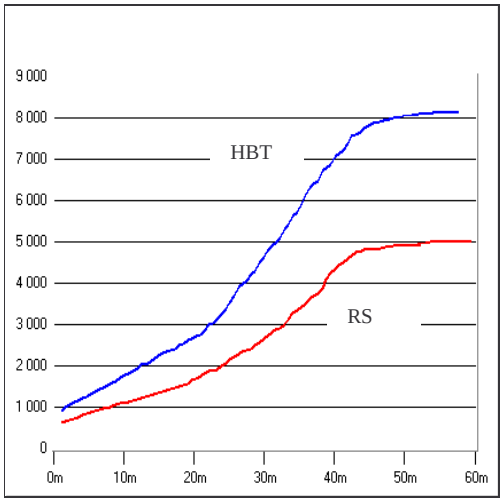


Figure 12: Number of Unicast packets

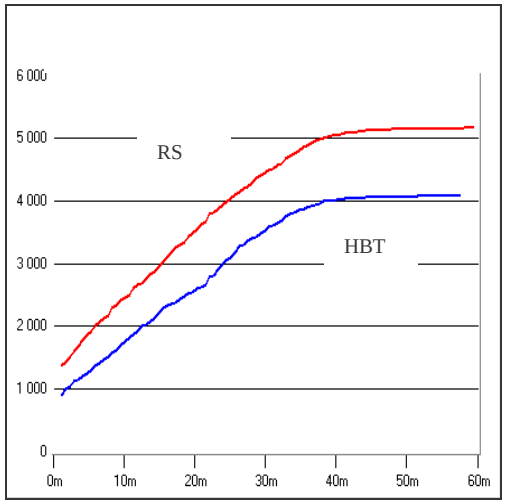


Figure 13: Number of Multicast packets

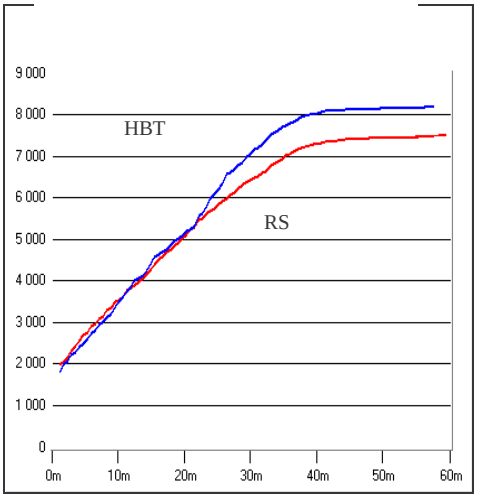


Figure 14: Number of packets on a join

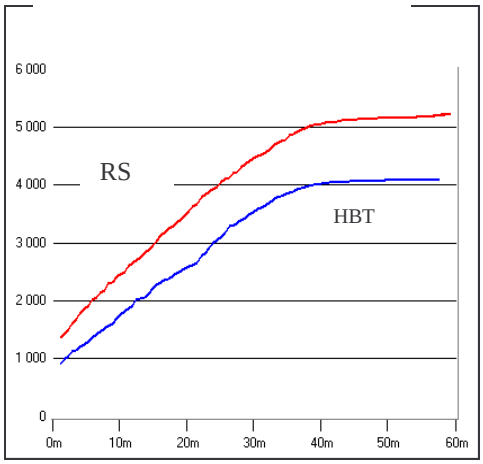


Figure 15: Number of Unicast packets in a join

The number of Unicast packets on a join is greater with our method. This is explained by the transmission of our identifiers to the new participant. The identifiers associated with will be sent to others in Multicast packets.

On a leave of a participant, the total number of packets exchanged is less with the RS method than with the HBT method. This leads to say that it produces less congestion on networks and produces less charge on the KDC.

The number of Unicast packets sent on a leave is 0 (as shown in figure 18). The participant willing to quit the group has no information to receive, with both methods. So, the number of packets on a leave is the number of Multicast packets (figure 19).

The identifiers in our method are sent on a join of a participant and therefore the Multicast packets can be reduced on a leave of a participant.

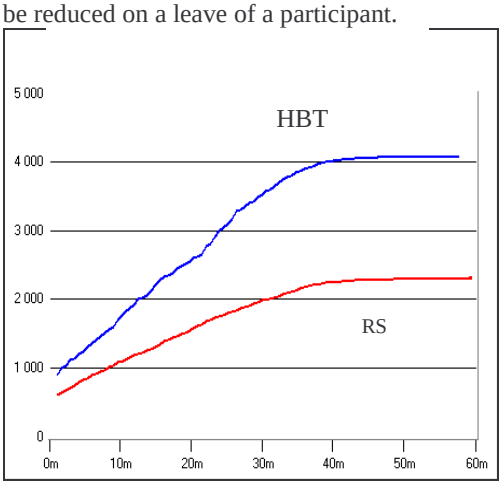


Figure 16: Number of Multicast packets on a join

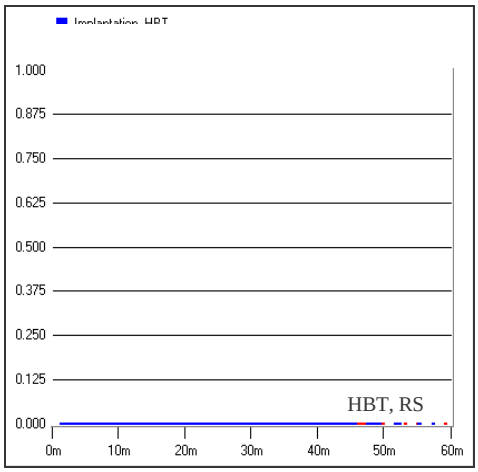


Figure 18: Number of Unicast packets on a leave

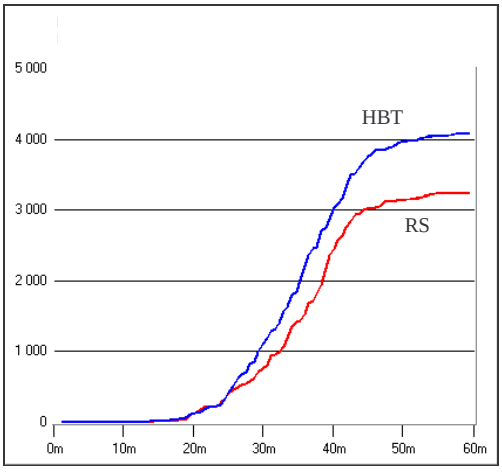


Figure 17: Number of packets on a leave

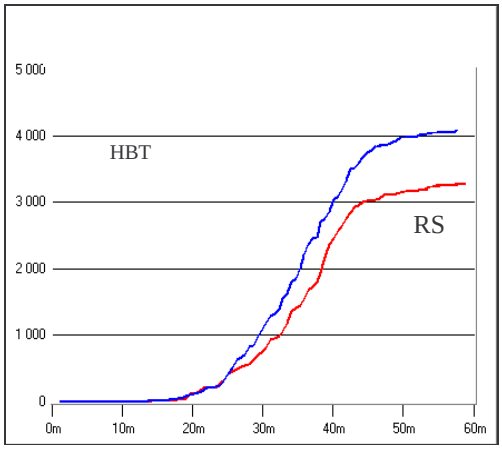


Figure 19: Number of Multicast packets on a leave

4.2.2.3 Computation time

When an encryption is made, the information is transformed. This information needs a lot of memory and computation time. When a method uses encryption it becomes even more complex.
With our method, we use keys already made and we modify them. This uses less time and less memory at the KDC.
The next figure (figure 20) shows the number of new encryptions made for the same group. The number of new encryptions is greater with the HBT method, so it confirms that it needs more resources and more computation time than the RS method.

On a join or a leave, less number of keys the new participant will know or the leaving participant knows have to be changed. Changing the values of these keys by a one-way function takes less computation time and less memory than with new encryptions.

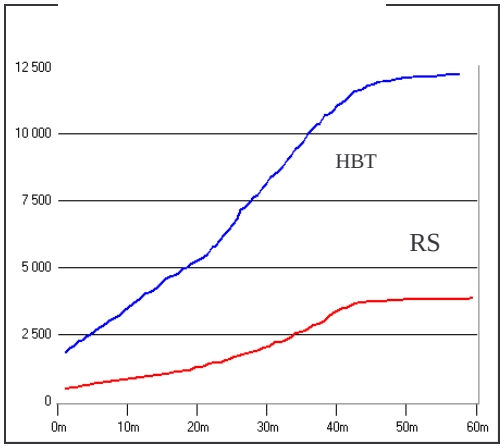


Figure 20: Number of new encryptions

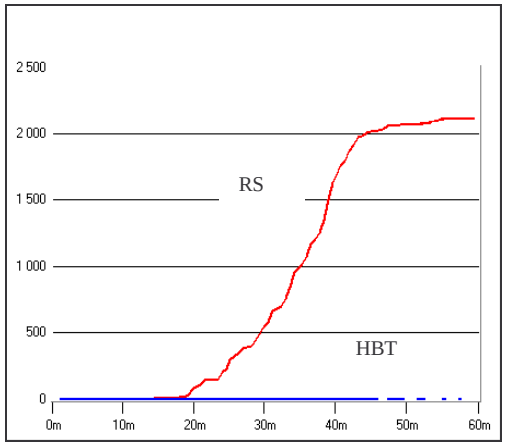
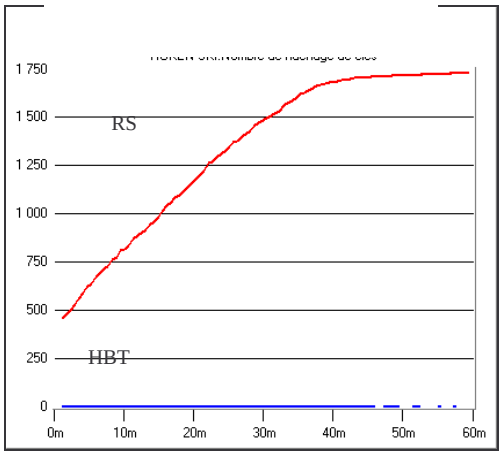


Figure 21: Number of key refreshes

The slided keys (or hashed keys) require the same amount of memory and computation than the refreshed keys; they are obtained by a second one-way function. The HBT method doesn't use those operations.



HBT

RS

Figure 22: Number of hashed key making

With the new encryptions requesting more resources, the HBT method requires more processing time than the RS method, which uses the modification of existing keys and demands less resources.
According to the next figures (figure 23, 24), the processing time of the RS method will be the lowest, because it require less keys to create. This is due to the amount of new encryptions and hashed keys.

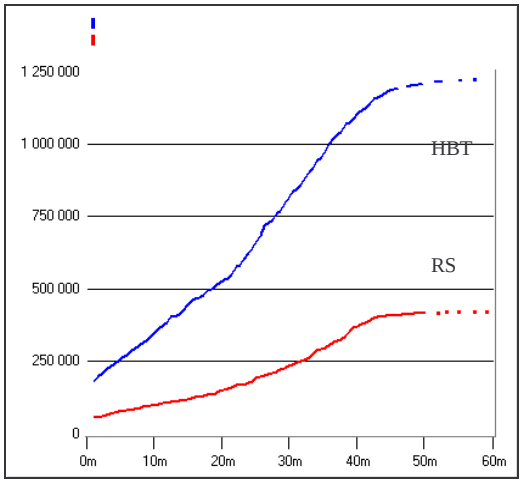


Figure 23: Processing time using a normal distribution

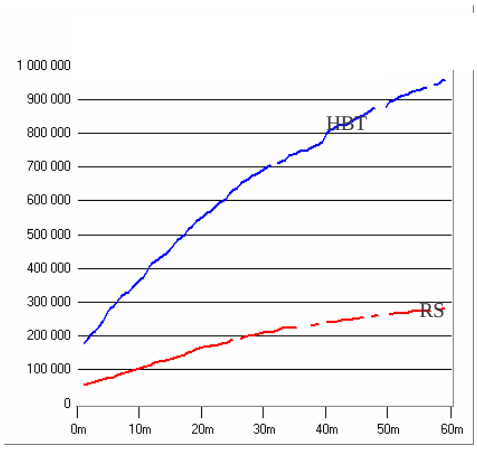


Figure 24: Processing time using an exponential distribution

5 Conclusion

We presented the general research trend during the last years regarding architectures for new keys' exchange techniques. The objective of these proposals was to bring an answer to the complex problem of keys' exchange in Multicast communications. These solutions are often partial due to the fact that they are specific to certain groups (size). Our work brings an additional stone to this building, a pragmatic solution based on the combination of the identifiers approach's technique with the slided keys method.

The RS approach tends to reduce the number of transmissions while guarantying the confidentiality and preventing collusion between participants of a group. The use of identifiers decreases by half the number of encryptions and key transmissions when the join and leave operations are combined.

By using the key naming method, the approach facilitates the comprehension of the tree for the individuals and also facilitates the treatment when refreshing keys. This new method can also be used to find which participant of the group knows a certain key. By this, the structures of hierarchy of keys are absent, so storage is also decreased.

In this work, we introduced the new RS method. Its goal was to normalize and reduce the packet exchanges on joins and leaves of participants. We have shown that the RS method is better than the HBT method. The quantity of packets can be lowered by a good ratio while preserving a low collusion probability. In addition to the packets reductions, we have shown that the RS method reduces the processing time at the KDC. The majority of the simulations results presented in this paper was done with a normal distribution. We also used an exponential distribution. The results obtained give the same tendencies and we arrive at the same conclusions: our approach is better than HBT method.

6 References

[1] Wallner, D. , Harder, E., Agee, R., "Key management for Multicast : issues and architecture", RFC2627, June 1999.

[2] Wong, C., Gouda, M. and S. Lam, "Secure Group Communications Using Key Graphs", Technical Report TR 97-23, Department of Computer Sciences, The University of Texas at Austin, November 1998.

[3] Sandro Rafaeli, Laurent Mathy and David Hutchison, “EHBT: an efficient protocol for group key management”, NGC 2001, LNCS 2233, pp. 159-171, 2001.

[4] Adrian Perrig, Dawn Song, and Doug Tygar, “ Elk, a new protocol for efficient large-group key distribution”. Proceedings of IEEE Symposium on Security and Privacy, May 2001.

[5] McGrew D, Sherman A. T., "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", Technical report no 0755, TIS Labs at Network associates, inc, Glenwood, MD, 1998.

[6] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor and Benny Pinkas, “Multicast Security: a taxonomy and some efficient construction”, In Proc of INFO-COM’99, volume 2, pages 708-716, New York, NY, USA, March 1999.

[7] R. Canetti, B. Pinkas, “A taxonomy of Multicast security issues”. http://www.ietf.org/internet-drafts/draft-canetti-secure-Multicast-taxonomy-00.txt, May 1998.

[8] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner, “ Efficient security for large and dynamic multicast groups”. Proceedings of the IEEE 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '98), June 1998.

[9] T. Hardjono, R. Canetti, M. Baugher, and m P. Dinsmore, “Secure Multicast: Problem Areas, Framework, and Building Blocks”. Internet Engineering Task Force, October 1999. Draft-irtfsmug-framework-00.txt.

[10] Molva, R. and Pannetrat, A. “Network security in the Multicast Framework”, Networking and tutorials, LNCS 2497, p.59-82, 2002.

[11] Maughan, D., Schertler, M. Schneider, M. and J.Turner, "Internet Security Association and Key Management Protocol”, RFC 2408, November 1998.

[12] Koh, Jong-Young an al, “Knowledge-Base Modeling and Simulation of Network Access Control Mecanisms Representing Security Policies”, ICICS 2002, LNCS 2513. PP.374-385, 2002.

[13] Banerjee, Suman and al, “Scalable Secure Group Communication Over IP Multicast”, IEEE Journal on selected areas in communications, VOL 20, NO8, October 2002.

[14] Martin St-Amand, Boucif Amar Bensaber, Dominique Seret, “A new Key Distribution Scheme for Secure Multicast Communications”. IADIS www/Internet conference, Portugal, October 2003.

[15] Martin St-Amand, Boucif Amar Bensaber , André Boumso, “ROKeN-SKI : A new pragmatic solution for key distribution in secure Multicast communication”, Medhoc Conference, Tunisie, June 2003.

7 Biography

Martin St-Amand is a Master student in Mathematical and computer sciences at Université du Québec à Trois-Rivières. His research interest includes Multicast networking and security.
Boucif Amar Bensaber received the PhD degree in computer sciences from René Descartes University, Paris V, France, in 1998. He is currently Professor, at the Mathematics and Computer Departement of Université du Québec à Trois-Rivières, Quebec, Canada. His research interest includes Multicast, Wireless local Network and security.
Ismail biskri has received the Ph.D degree in computer science from LaLIC/EHESS/CNRS/ Paris-Sorbonne in 1995. He is currently professor, at the Mathematics and Computer science Department at *Université du Québec à Trois-Rivières, Québec, Canada*. His research interest includes artificial intelligence, data-mining.
Abdelhamid Zebdi is a Master student in Mathematical and computer sciences at Université du Québec à Trois Rivières. His research interest includes Multicast, wireless and routing.
Dominique Seret is a Vice-Dean of the René Descartes University, in charge of the Information and Communication Technology. She is Professor and Dean of the Mathematics and Computer Science Department. Her research field is Computer Network and Quality of Service.

This work was completed with the support of the Natural Science and Engineering Research Council of Canada (NSERC).