# SMIz - Secure Multicast IPTV with efficient support for video channel *zapping*

António Pinto
Escola Superior de Tecnologia e Gestão de Felgueiras, Politécnico do Porto
INESC Porto, Portugal
apinto@inescporto.pt

Manuel Ricardo
INESC Porto, Faculdade de Engenharia, Universidade do Porto, Portugal
mricardo@inescporto.pt

June 22, 2008

## Abstract

The current trend in technological evolution is leading telecommunications to all-IP network scenarios, where multiple services are transported as IP packets. Among these are the services based in group communications over IP with confidentiality requirements. Secure IP multicast may be used for the secure broadcasting of multiple video channels over IP. However, scenarios where users switch rapidly between video channels (*zapping*) introduce new requirements, including very low channel access times and low amounts of signaling.

The solution proposed in this paper addresses these requirements, without neglecting user access control. For that purpose, a centralized form of secure group communication is used to transmit, not data, but group cryptographic material that will enable end systems to decrypt the group communication, while satisfying these new requirements.

## 1 Introduction

The technological evolution is leading telecommunications to all-IP network scenarios, where multiple services are transported as IP packets. Solutions for transmitting multimedia contents in IP packets already exist, and the Real-time Transport Protocol (RTP) is a key component of these solutions. Among these are the services based in group communications over IP with confidentiality requirements, such as television, video on demand, and video conferencing. These services triggered research activities in group communications envisaging two main topics: optimized transmission and secure communication. Secure IP multicast [1] may be used for the secure broadcasting of multiple channels over IP (e.g. IPTV), but aspects such as the transmission of multiple real-time video channels which allow users to switch rapidly between them (channel surfing or zapping) still need to be addressed in scenarios consisting of large multicast groups, where the adaptability of secure multicast may be compromised.

Several proposals exist in literature for providing scalable secure group communications [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. However, no secure group communication protocol is adequate to all scenarios. Scenarios demanding more research work include the management of highly dynamic groups, the management of groups of the mobile users, scenarios requiring low computation on cryptographic operations at end-systems, or scenarios requiring the adaptation of data or cryptographic techniques to heterogeneous networks.

The solution proposed in this paper addresses the reduction of signaling in scenarios where users rapidly switch between video channels in a multiple video distribution IPTV service without neglecting the user access control. This solution can be characterized as a centralized secure group communication used to transmit, not data, but group cryptographic material that will be used by end systems to decrypt the group communication data.

The paper is structured in four chapters. Chapter 2 presents the work which is related to ours. Chapter 3 presents our solution. Chapter 4 presents the performance results of our solution obtained both by simulation and by using a testbed. Chapter 5 presents the conclusions of the work.

## 2 Related work

Secure multicast is a group transmission technique that enforces confidentiality. It uses cryptographic techniques to encrypt data and perform access control. A secure multicast architecture needs to consider the size of the groups, group memberships, and security contexts such as encryption keys. Architectures such as those defined in [12] are efficient for small groups, where the architecture defined by the Multicast Security (MSEC) group [13] is being developed for large groups [14, 15].

In its simplest scheme, the source of the group sends data to an IP multicast address; a receiver interested in the data signals its interest to its local multicast router using a IGMP [16] or a MLD [17] join message. Access control is imposed by encrypting the data prior to its transmission, and by sending the decryption key to the authorized receivers. Group confidentiality is obtained by changing the decryption key, and by transmitting it securely to the authorized receivers. Decryption keys can be transmitted regularly, upon a group change, or using a combination of both methods. Group changes occur either by the departure of a member (group leave) or by the arrival of a new member (group join). The operations of key renewal are referred in the literature as re-key operations, and are usually managed by an entity named Group Controller (GC).

Some types of cryptographic keys are used in secure group communication architectures. The common types are Key Encryption Keys (KEK), and Data Encryption Keys (DEK). KEK is a key assigned to a member which is known only by the member and the GC; the KEK is used to secure communications between each member and its GC. A DEK is a key used to encrypt the group communications data and must be known by all the group members. In a re-key operation, for instance, the DEK may be securely transmitted by the GC to valid members in a message composed of $[\{DEK\}_{KEK_1}, \{DEK\}_{KEK_2}, ..., \{DEK\}_{KEK_n}]$. In this example, the notation $\{DEK\}_{KEK_1}$ means that the DEK is encrypted with the KEK of the first member.

Confidentiality requirements can be classified in four classes [2]: 1) non-group confidentiality; 2) forward secrecy; 3) backward secrecy; 4) collusion resistance. The first class imposes that users that never participated in the group should not access any cryptographic material. The second class of requirements imposes that a member departing from a group should stop receiving cryptographic material, therefore ensuring that this member is unable to decrypt group communications after leaving the group. The third class imposes that a receiver arriving to the group should not access previous cryptographic material, ensuring that this member is unable to decrypt past group communications. The last class imposes that current cryptographic material should not be inferable by non-members.

In [18] three approaches for key distribution were identified: centralized, distributed, and decentralized. More recently, Cao et al. [3] extended this classification and identified four schemes: simplest scheme, centralized scheme, decentralized scheme, and hierarchical scheme. The first scheme is a subset of the centralized approach; the centralized and the decentralized schemes are the centralized and the decentralized approaches respectively. In this paper we adopted a classification that combines both and comprises 4 types of key distribution: centralized, decentralized, distributed and hierarchical.

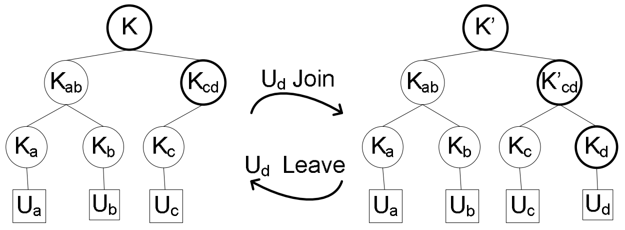

Figure 1: KEKs affected by member $U_d$ join/leave.

## 2.1 Centralized type

The centralized type is characterized by the existence of a unique entity which manages the entire group; it aims at optimizing the bandwidth consumption, minimizing the key storage space, and reducing the computational power. The GC encrypts the DEK using each member's key ($KEK_i$) and then it transmits the $n$ keys to the group members. Despite its simplicity, this scheme suffers from the single point of failure problem; in case of failure of the GC, the cryptographic material is not renewed and the new members become unable to receive the cryptographic material required to decrypt the data.

In order to address problems such as key storage and the support of highly dynamic groups, the GC maintains the keys in the from of a balanced tree whose leafs are the individual member KEKs and the intermediate nodes represent the KEKs required by the underlying nodes. An example tree is shown in Figure 1. The root of the tree holds the group DEK ($K$). When a new member joins the tree, it is added as a leaf to the tree and all the keys in the path from its parent node to the root are changed. These keys will then be used by the new member to obtain the group key, i.e. the root of the tree. The groups with high rates of member departure and arrival can be supported by using these trees, since only the affected keys are refreshed. Examples of centralized schemes are LKH [4, 19] and key graphs [19].

The join operation of the member $U_d$, shown in the Figure 1, requires several encryptions in order to maintain backward secrecy. The key $K'$ becomes the new root DEK and it must be sent to all members. The key $K'_{cd}$ must be sent to the members $U_c$ and $U_d$. Thus, the KEK refresh message shall contain: 1) a copy of $K'$ encrypted with $K_{ab}$, $K_c$, and $K_d$; 2) a copy of $K'_{cd}$ encrypted with $K_c$ and $K_d$. The member $U_c$, for instance, receives $\{K'\}_{K_c}$ and $\{K'_{cd}\}_{K_c}$, while members $U_a$ and $U_b$ receive the same message $\{K'\}_{K_{ab}}$. This method generates a high number of re-keying messages; for a group of $n$ users, $O(log\, n)$ re-keying messages are required and each member needs to store $O(log\, n)$ keys; the number of keys stored by each user is equal to its distance to the root of the tree.

In [5] the One-way Functions Tree (OFT) was introduced and it enabled users to derive the new KEK from the current KEK. Upon a group change, each member must derive the new key for each node in the path from its parent's node to the root by using One-way Functions. Such strategy allows a reduction of the number of re-keying messages to half, but it substitutes the message cost by a computational cost. Canneti et al. [20] introduced a solution which uses pseudo-random number generators instead of OFT in order to derive the new KEK from the current KEK. ELK [6] focuses on large groups and uses pseudo random functions to derive new keys instead of OFT; besides, it periodically updates all the keys of the tree regardless of group membership changes. In [21], LKH++ was proposed which exploits one-way hash functions in combination with information already shared by the users. In particular, upon a user leave operation, the affected users may autonomously generate the new keys in the path towards the root, thus reducing the number of re-keying messages.

## 2.2 Decentralized type

The centralized type assumes that each member has no knowledge of other group members and that the access control is performed by a unique entity. In the decentralized type, the group is split into subgroups, each having its manager. The subgroup manager generates the local encryption key and processes the local membership changes (subgroup member join/leave operations). Iolus [7] is a decentralized scheme where each subgroup is managed by a Group Security Agent (GSA). GSAs are hierarchically organized and form a top-level group managed by the Group Security Controller (GSC), as shown in Figure 2. Iolus requires that the GSC trust the GSAs. In Iolus, the keys used

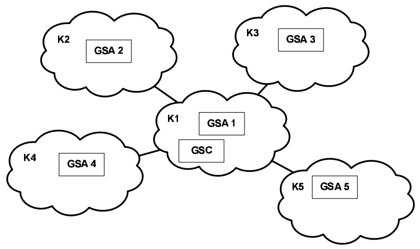

Figure 2: Iolus framework

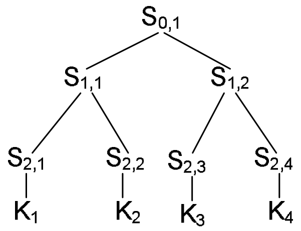

Figure 3: MARKS Binary hash tree

by each subgroup are independent, what enables membership changes to be treated locally. Key independence implies that the GSA must be in the data path so that it can decrypt the original data and re-encrypt it using its local subgroup key. In case of failure of a GSA, only the subgroup it manages is affected. Despite the scalability advantage when compared to the centralized type, the Iolus' GSAs is itself a bottleneck because it decrypts and encrypts all the data for the group it manages.

Dondeti et al. proposed DEP [8]. Here, the element responsible for subgroup management is named Subgroup Manager (SGM) and it is similar to the Iolus' GSA. The scenario adopted considers the deployment of SGMs in third party equipments such as Service Provider (SP) routers, what demands a data and key distribution scheme which avoid group data disclosure to these SGMs. DEP classifies SGMs as member and participant. The member SGMs are entitled to access both DEP and group data, where the participant SGMs are not. DEP uses one DEK and two types of KEK: 1) KEKs known by the GC, senders and members; 2) Local Subgroup keys (LSs), known by members and participant SGMs. The DEK, required by all members in order to decrypt the group communications, is sent by the GC to the SGMs in the form $\{DEK\}_{KEK_i}$. In turn, the participant SGM encrypts the encrypted DEK with its $LS_i$, and transmits $\{\{DEK\}_{KEK_i}\}_{LS_i}$ to its subgroup members. Every member is now able to decrypt the double encrypted DEK. Despite involving trusted third parties, this solution does not enable participant SGMs to access DEK but it still enables them to process subgroup membership changes by refreshing their $LS_i$. This solution has two drawbacks: it requires extra encryption/decryption operations, and the leaving members, in possession of the current DEK, will be able to access the group data until the DEK is refreshed.

In contrast with the session oriented group keys, MARKS [22] associates seeds to time slices of group communications, each time slice corresponding to a key. The seeds form a binary hash tree, where leaves correspond to time slices; the seeds in the path to the root are required in order to generate the time slice cryptographic key. Figure 3 shows a scenario with four time slices and their cryptographic keys ($K_1$, $K_2$, $K_3$, $K_4$). If an user wants to access the first time slice, he will need seed $S_{2,1}$; if he also wants to access time slices 3 and 4, he will need to obtain seed $S_{1,2}$. In the possession of seed $S_{1,2}$, every member is able to generate seeds $S_{2,3}$ and $S_{2,4}$.

The intra-domain group key management (IGKMP) [23] assumes 2 main entities: the Area Key Distributor (AKD), and the Domain Key Distributor (DKD). Each AKD manages an area subgroup. The DKD is responsible by the group key generation and their transmission to AKDs which, in turn, send the group key to their area members. All the key distributors form a multicast group that is used in key refresh operations. The group key is the same in all areas, therefore no decryption and re-encryption is required in group communications from one area to the other.

Kronos [10] was proposed by Setia et al. and focuses in periodic batch key refreshing, neglecting membership changes during each period. A new group key is generated and distributed after a period of time without considering inner period member join and leave operations. There are subgroup managers (AKDs), that can independently generate the same group key and do not demand its transmission from the DKD. In order to generate the same group key, the AKDs need a synchronized clock and have to agree in two secret factors, which are used to derive the first key. The following keys are derived by encrypting the current key with the master key which is one of the two secret factors. Because all the keys are derived from the current key, security may be compromised if one key is disclosed. This decentralized approach requires mechanisms such as clock synchronization and conflict resolution.

## 2.3 Distributed type

The distributed type assumes that every member can participate in the key distribution, perform access control, and contribute to the generation of the group key. The group controller role is not usually present because the group keys are generated with contributions from all the members. Group Diffie-Hellman Key Exchange [24] is an example of the distributed scheme. It consists of the extension of the Diffie-Hellman key agreement protocol to groups of users. A group of $n$ members firstly agrees on a pair of prime numbers ($q$ and $\alpha$). Each member generates also its secret number ($s_i$). The first member must calculate $\alpha^{s_1}$, as its intermediate value, and sends it to the next member. Each subsequent member $i$ will, in turn, generate a set comprising $i$ intermediate values with $i-1$ exponents, plus a cardinal value containing all exponents. The fourth member, for instance, receives the set: $[\alpha^{s_2 s_3}, \alpha^{s_1 s_3}, \alpha^{s_1 s_2}, \alpha^{s_1 s_2 s_3}]$, and transmits to the fifth member: $[\alpha^{s_2 s_3 s_4}, \alpha^{s_1 s_3 s_4}, \alpha^{s_1 s_2 s_4}, \alpha^{s_1 s_2 s_3}, \alpha^{s_1 s_2 s_3 s_4}]$. The cardinal value of a member is the last value of the set generated; in this example $\alpha^{s_1 s_2 s_3 s_4}$ is the fourth member's cardinal value. The nth member can obtain the group key $k$ by calculating: ($k = \alpha^{s_1 \cdots s_n} \, mod \, q$). Since all the members contribute to the group key, the processing time and network resources usage of this scheme increases linearly with the group size.

## 2.4 Hierarchical type

The hierarchical type assumes that users have not the same priorities and impose cryptographic access control for classes of users with different access levels. This scheme was firstly addressed in [25] where a hierarchical key assignment to users was adopted. An user belonging to a certain class can derive the cryptographic keys of lower classes.

| | Number of Keys | | Re-key message size | |
|---|---|---|---|---|
| | All | Member | Join | Leave |
| Simplest | $n$ | 1 | $n$ | $n$ |
| LKH | $2n-1$ | $h+1$ | $(2h-1)+(h+1)$ | $2h$ |
| LKH++ | $2n-1$ | $h+1$ | $h+1$ | $h+1$ |
| ELK | $2n-1$ | $h+1$ | $h+1$ | $h$ |
| OFT | $2n-1$ | $h+1$ | $2(h+1)$ | $h+1$ |

Table 1: Centralized approaches comparison

Nevertheless, the hierarchical scheme presents the drawback of requiring extra computational power at the members, similarly to the distributed scheme.

## 2.5 Summary on related work

Table 1 compares the centralized approaches identified in this paper. In this table, $n$ represents the number of members in the group and $h$ represents the height of the tree used to maintain the keys in the GC. The $1^{st}$ column shows the number of keys maintained by the group controller. The $2^{nd}$ column shows the number of keys required by each member. The $3^{rd}$ and $4^{th}$ columns show the message size in numbers of keys that must be transmitted upon group memberships changes in order to preserve secrecy, respectively in join and leave operations. For the *simplest* approach, the group controller maintains one key per each member in the group; each member requires only one key, and both the group join and leave operations require the transmission of a re-key message with size $n$ times the key length (one key per member). The remaining solutions in the table, when compared with the simplest approach, show reductions in the bandwidth required for re-key operations upon membership changes.

The distributed type assumes that all members can contribute to the generation of the DEK and perform access control. The hierarchical type assumes that the users are not equal in terms of access, and are hierarchically organized into classes of users with different access rights. The decentralized splits the group into subgroups, each having a subgroup manager.

## 3 Proposed solution

The scenario adopted in the this work is an IPTV service, where multiple video channels are distributed as IP packets in multicast. In common IPTV services, multiple video channels are grouped together in bundles. The video channels are generated by one or more Video Servers (VS) to groups of Set-Top Boxes (STB). The requirements identified for this system include:

1. Individual user access control;
2. Support for legacy end-systems;
3. Transparent operation over existing networks and network equipments;
4. Low network resources consumption;
5. Support for rapid switching between video channels;
6. Scalability.

The first requirement imposes that a user must not derive or obtain the cryptographic materials of other users. The second requirement imposes that a solution must demand low computational power to both STBs and VS, in what concerns group security. The third requirement imposes that a solution must not require changes to existing networks or network equipments. The fourth requirement implies that minimal bandwidth shall be used for signaling. The fifth requirement imposes that users must be able to switch rapidly between video channels, similarly to what happens in analogue cable TV. The last requirement imposes that a solution must support a large number of users without significant impact on the system's performance.

The proposed solution, named Secure Multicast IPTV with efficient support for video channel *zapping*(SMIz), is depicted in Figure 4 and it comprises a Group Controller (GC), a Video Server, and a Set-Top Box. The GC is responsible for cryptographic key generation and distribution, as well as STB authentication and authorization. The VS and the STB are responsible for the stream transformation, i.e. for the encryption and decryption of the video channel streams. The VS transforms

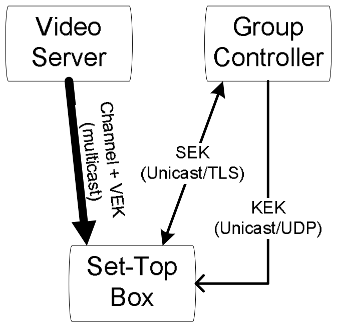

Figure 4: Proposed solution elements

the audio and video content into an encrypted stream of IP multicast packets; it also generates the Video Encryption Keys (VEKs) and distributes them to valid members (VEK announce). Prior to video channel request and visualization, the STB must obtain its cryptographic context.

Three types of cryptographic keys are used in this solution: 1) Session Encryption Keys (SEKs); 2) Key Encryption Keys (KEKs); 3) Video Encryption Keys (VEKs). SEKs are used for securing unicast communications between STBs and the GC. VEKs are used to (de)encrypt video channels. KEKs are used for securing the transmission of the VEKs.

In Figure 5 we show a message sequence chart which starts when the STB is turned on. At bootstrap, the STB establishes a secure unicast connection with the GC, using the Transport Layer Security [26, 27]. This connection is used to request the STB cryptographic context, which consists of a Session Encryption Key (SEK), the cryptographic algorithm, and the key size. This key (SEK) will be used to secure the video channel requests sent by the STB to the GC, enabling the STB to securely obtain the current KEK for the requested video channel. In order to receive the multicast transmission of the requested video channel, the STB must also send an IGMP join message to its designated multicast router. The destination address of this

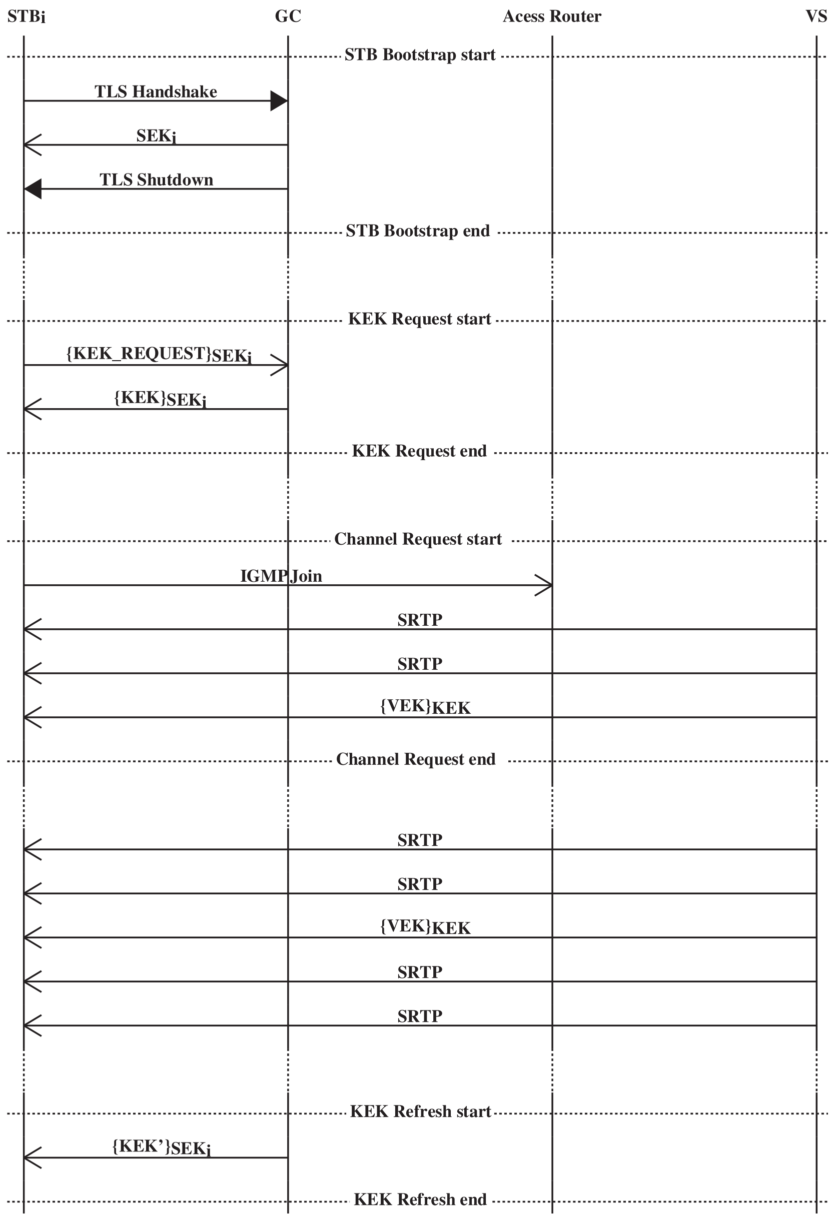

Figure 5: Message sequence chart

join request is the group address assigned to the video channel the user wants to receive. Each video channel is transmitted to its multicast group address in the form of Secure Real-time Transport Protocol (SRTP) packets, encrypted with a VEK. The VEKs are sent periodically with each video channel stream, to the same IP multicast group address, but to a different UDP port number; they are encrypted with the KEK. The STB decrypts the VEK with the KEK, and then decrypts the video channel stream with the VEK. To ensure a high level of security, all cryptographic keys must be refreshed periodically. These key refresh operations (re-keys) must not interfere with the video channel visualization of current receivers. Thus, each VEK is associated to a maximum SRTP packet sequence number, after which the VEK is no longer valid. The SEK is renewed upon each STB bootstrap and the KEKs are sent periodically by the GC to all STB, prior to their expiration. As a fall back procedure, the STB is also able to request the current KEK.

The support of fast switching between multiple video channels is achieved by the adoption of periodical VEK announces, transmitted from the VS to the same group address of the video channel. VEK announces are secured by $\{VEK\}_{KEK}$ and only one VEK announce is needed for all members, since all of them share the same KEK. This procedure leads to savings in signaling, specially because VEKs are the keys which are re-keyed more frequently.

## 4 Results

In order to validate the solution proposed, a prototype was developed and implemented in a testbed; the results obtained from the functional and performance tests are presented in Section 4.1. Then, in Section 4.2, some simulation results regarding the cost of the KEK re-key operations, with respect to both group size and re-key interval, are presented. In Section 4.3, a comparison with other solutions is performed in what concerns the network bandwidth used for signaling in scenarios where users zapp through video channels.

In the prototype, 3 type of cryptographic keys where used, each with a different security level. SEKs (256 bit keys) are the most secure because they are the STB individual keys. KEKs (192 bit keys) are

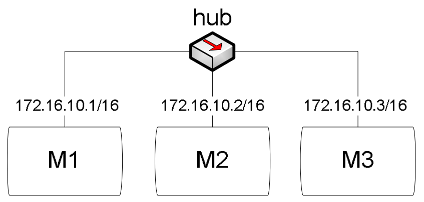

Figure 6: Test bed used in the experimental results

Table 2: KEK Requests processed per second

| | Average | Std. Dev. |
|---|---|---|
| AES-128 | 20814 | 345 |
| AES-192 | 20121 | 356 |
| AES-256 | 19308 | 754 |

shared between all STBs that subscribe the same set of video channels and demand a security level higher then VEKs. As the number of times a STB decrypts packets with KEKs is higher than the number of times STB decrypts packets with SEK, and in order to reduce the computational power required, the KEKs are made shorter then the SEKs. The VEKs (128 bit keys) are shared only by the STBs which receive the same video channel at each moment. Because different key sizes enforce different security levels, these keys must be refreshed with different time intervals. The VEKs are refreshed more often than KEKs, and KEKs are refreshed more often than SEKs.

## 4.1 Experimental results

The testbed implemented and used to obtain experimental results is represented in Figure 6. It consists of 3 computers interconnected through an Ethernet hub. All the computers have the same hardware and software characteristics, namely the Fedora Core Linux operating system, an 3000+ AMD Athlon 64 processor and 1GB of RAM memory.

The first experimental test focused on the STB bootstrap procedure

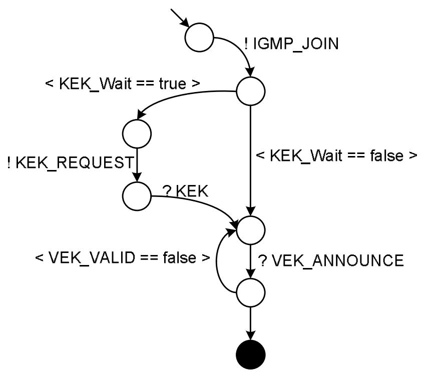

Figure 7: Third experiment state machine

Table 3: VEK processing time (ms)

| VEK Re-key interval (ms) | VEK Processing time | |
|---|---|---|
| | Average | Std. Dev. |
| 500 | 972.79 ms | 3.61 ms |
| 100 | 131.66 ms | 3.70 ms |

(see Figure 5). It consists in the establishment of a TLS secured TCP connection with the GC in order to obtain the SEK, algorithms and key sizes. The GC, in M1, was configured with a 200,000 STB database, which was loaded to RAM memory in the form of an hash table with approximately 10,000 buckets. The memory usage for the 200,000 STB was 256MB. 10 instances of STB where executed in M2, repeatedly requesting their session cryptographic contexts. M3 was used to capture network traffic in order to evaluate the network usage. As a result, the GC was able to correctly process an average 212 requests per second, using 1.35 Mbit/s of network bandwidth.

The second experimental test focused on stressing the server regarding the KEK Request procedure. This procedure, also introduced in Figure 5, consists on the reception of the KEK request message by the GC, extraction of STB identification, request validation and authenti-

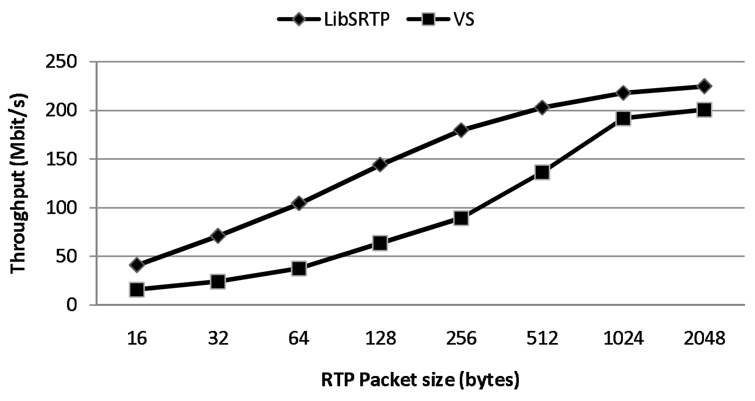

Figure 8: Video Server throughput

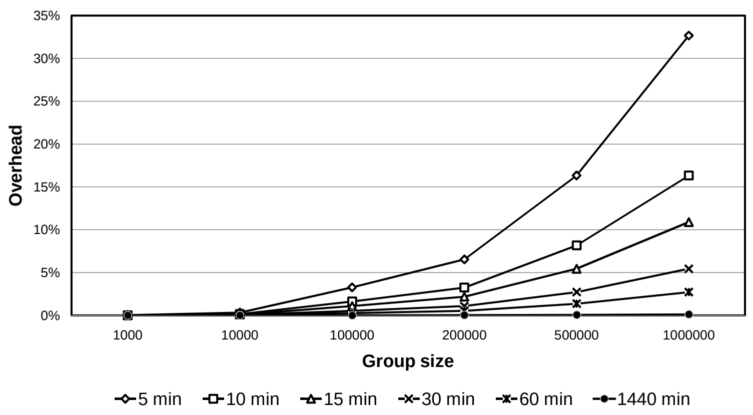

Figure 9: Bandwidth overhead, in percentage, per video channel KEK refresh

cation, and, upon successful authentication, construction and transmission of the KEK reply message. The STB was setup in M2 in order to send cyclic KEK requests. The GC was setup in M1. Each test was repeated multiple times using different key sizes of the Advanced Encryption Standard (AES) algorithm. Table 2 shows the results obtained for the hash table worst case bucket offset. The average time for a STB to obtain a KEK reply was 7ms.

The third experiment focused on the time required by a STB to obtain both KEK and VEK, which are both required to decrypt a video channel, with respect to VEK announce intervals of 500 and 100 ms. A simplified state machine of this procedure in $STB_i$ is shown in Figure 7 and it assumes that the STB is already in possession of its SEK. Here, ! and ? represent respectively the transmission and the reception of messages. The GC was setup in M2, the STB was setup in M3, and the VS in M1. Each test was executed 8 times per VEK re-key interval. The results obtained are presented in Table 3, where the values represent the time, in ms, until the STB receives a valid VEK.

The last experiment aimed at evaluating the VS overhead introduced by the proposed solution. For that purpose, the throughput of the native libSRTB was measured and compared with the throughput obtained using our solution. The results obtained are shown in Figure 8 and they show that, for typical packets of 1500 bytes, the differences of throughput are of about 10%.

## 4.2 Simulation results

The KEK refresh operation is the operation which may affect the performance of our solution, since it is carried as an unicast UDP packet per subscriber and per KEK re-key interval; the bandwidth required for KEK re-key grows linearly with the group size. Figure 9 presents the cost in terms of network resources of the KEK refresh operation, normalized to a video channel bandwidth (4 Mbit/s). Results are shown for a group size of up to one million members and for KEK re-key intervals of 5, 10, 15, 30, 60 and 1440 minutes. For a group size of one million members, and for a KEK re-key interval of one day (1440 minutes), the bandwidth usage will be of 0.1% of a video channel.

## 4.3 Comparison

Figure 10 compares our solution with the centralized solutions identified in literature that require less bandwidth in signaling, the main goal

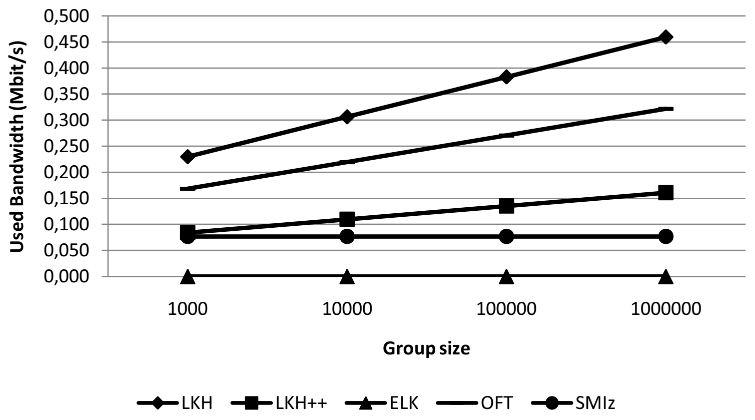

Figure 10: Bandwidth used in VEK re-key operations

of our solution. This comparison is done in terms of the bandwidth used in VEK re-key operations during video channel visualization. The bandwidth used in the member's bootstrap and the first group join operations are not considered. The SMIz and ELK solutions lead to a constant bandwidth usage or, in other words, the required bandwidth does not grow with group size. ELK does not demand traffic in this situation because the new keys are obtained by each member by computing Pseudo-Random Functions (PRF) over the tree information in their possession. SMIz does not require multiple PRF computation at each member per refreshed key and it enables key independence, since future keys do not depend on previous keys.

Figures 11(a) to 11(d) show the bandwidth spent in signaling generated by zapping channels. It is represented as a function of group size and percentage of members zapping through the multiple video channels. In Figure 11(b), for instance, a group of one million members, of which 30% are switching channels, consume approximately 803 Mbit/s in this signaling. The bandwidth required for signaling in all solutions, except ours, grows with group size. It was assumed a tree height equal to $log_2(n)$, $n$ being the group size.

Our solution requires less bandwidth in all the scenarios discussed,

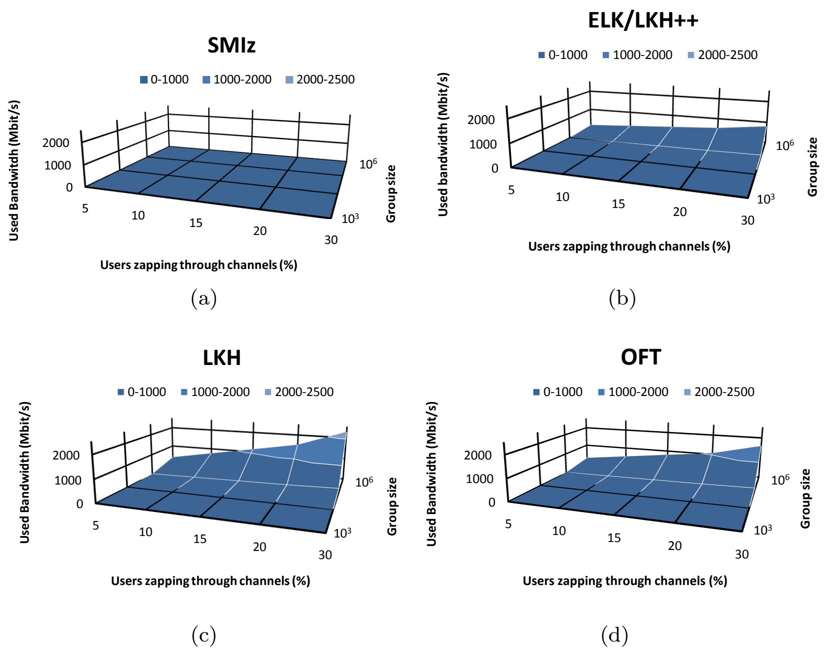

Figure 11: Signaling represented as a function of group size and percentage of *zapping* members

except for the case of having only 5% of the users zapping through video channels for a group size of one thousand. In this case, our solution requires 80 kbit/s, where ELK or LKH++ require 70 kbit/s. While LKH, LKH++, OFT and our solution require signaling in re-key operations, ELK does not. In our scenario each video channel is transmitted to a different group address, so switching from one channel to another implies a group leave operation (from the current channel) and a group join operation (to the new channel). All mentioned solutions require KEK related signaling in group join and leave operations, except ours.

## 5 Conclusion

In this paper we proposed a centralized secure group communication solution used to transmit, not data, but group cryptographic material that will be used by members to decrypt the group data. The solution's main advantage over other solutions is the low and constant use of bandwidth for signaling in zapping scenarios, considering multiple real-time video channel. Moreover, our solution has proved to not significantly impact on the time required for group join operations. The prototype implemented has shown to scale up to one million of users.

Enabling users to switch rapidly between video channels in a multiple real-time video channels IPTV service (*zapping*) was the central issue of this paper, so the scalability problem was left for future work. Scalability can be reached by decentralizing our solution.

## References

[1] A. Pinto and M. Ricardo, "Multicast deflector," *Telecommunication Systems*, vol. 37, pp. 145–156, Apr. 2008.

[2] Y. Challal, H. Bettahar, and A. Bouabdallah, "Sakm: a scalable and adaptive key management approach for multicast communications," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 55–70, 2004.

[3] J. Cao, L. Liao, and G. Wang, "Scalable key management for secure multicast communication in the mobile environment," *Pervasive and Mobile Computing*, vol. 2, pp. 187–203, Apr. 2006.

[4] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Netw*, vol. 8, pp. 16–30, 2000.

[5] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The versakey framework: versatile group key management," *Selected Areas in Communications, IEEE Journal on*, vol. 17, pp. 1614–1631, 1999.

[6] A. Perrig, D. Song, and D. Tygar, "Elk, a new protocol for efficient large-group key distribution," in *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, 2001, pp. 247–262.

[7] S. Mittra, "Iolus: a framework for scalable secure multicast," in *Proc. of ACM SIGCOMM'97.* Cannes, France, 1997, pp. 277–288.

[8] L. Dondeti, S. Mukherjee, and A. Samal, "Scalable secure one-to-many group communication using dual encryption," *Computer Communications*, vol. 23, pp. 1681–1701, 2000.

[9] B. Briscoe, "Marks: Zero side effect multicast key management using arbitrarily revealed key sequences," in *Proc. of First International COST264 Workshop on Networked Communication*, 1999, pp. 301–320.

[10] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: a scalable group re-keying approach for secure multicast," in *Proc. of 2000 IEEE Symposium on Security and Privacy*, 2000, pp. 215–228.

[11] S. Rafaeli and D. Hutchison, "Hydra: A decentralized group key management," in *Proc. of 11th IEEE International Workshops on Enabling Technologies: Infrastructure for Colaborative Enterprises (WETICE'02)*, Los Alamitos, California, 2002.

[12] M. Steiner, G. Tsudik, and M. Waidner, "Cliques: A new approach to group key agreement," in *Proc. of IEEE ICDCS'98*, 1998.

[13] "Multicast security (msec)," IETF Working Group. [Online]. Available: http://www.ietf.org/html.charters/msec-charter.html

[14] T. Hardjono and B. Weis, *The Multicast Group Security Architecture.* RFC 3740, 2004.

[15] M. Baugher, R. Canetti, and L. Dondeti, *Multicast Security (MSEC) Group Key Management Architecture.* RFC 4046, 2005.

[16] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, *Internet Group Management Protocol, Version 3.* RFC 3376, 2002.

[17] S. Deering, W. Fenner, and B. Haberman, *Multicast Listener Discovery (MLD) for IPv6.* RFC 2710, 1999.

[18] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 309–329, 2003.

[19] H. Harney and E. Harder, "Logical key hierarchy protocol," *draft-harney-sparta-lkhp-sec-00. txt, IETF Internet Draft (work in progress)*, 1999.

[20] Canetti, Malkin, and Nissim, "Efficient communication-storage tradeoffs for multicast encryption," 1999.

[21] R. D. Pietro, L. Mancini, and S. Jajodia, "Efficient and secure keys management for wireless mobile communications," *Proceedings of the second ACM international workshop on Principles of mobile computing*, pp. 66–73, 2002.

[22] B. Briscoe, "Marks: Zero side effect multicast key management using arbitrarily revealed key sequences." Springer-Verlag, 1999, pp. 301–320.

[23] T. Hardjono, B. Cain, and I. Monga, "Intra-domain group key management protocol (igkmp)."

[24] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-hellman key distribution extended to group communication." New Delhi, India: ACM, 1996, pp. 31–37.

[25] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Trans. Comput. Syst*, vol. 1, pp. 239–248, 1983.

[26] T. Dierks and C. Allen, *The TLS Protocol Version 1.0*. RFC 2246, 1999.

[27] T. Dierks and E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.1*. RFC 4346, 2006.

**António Pinto** received his Licenticatura (2000) in computer science from Polytechnic of Porto, and M.Sc. (2005) degrees in communication networks and services from Porto University. Currently, he is an assistant professor at Escola Superior de Tecnologia e Gestão de Felgueiras of the Polythecnic of Porto, where he gives courses in computer networks. He is also currently involved in a PhD program on secure multicast services at Porto University.

**Manuel Ricardo** received a Licenticatura (1988), M.Sc. (1992), and Ph.D. (2000) degrees in Electrical and Computer Engineering from Porto University. Currently, he is an associate professor at the Faculty of Engineering, Porto University, where he gives courses in mobile communications and computer networks. He also leads Wireless and Mobile Networks area of INESC Porto.