

A Key Management Solution for Overlay-Live-Streaming

Anonymous
Some Research Group
Some Institution
Some Email Addresses

Abstract—Confidential communication of live-generated multimedia data distributed via application level multicast (ALM) still remains a mostly unaddressed subject even though some important usage scenarios, e.g. paid subscription services or personal video-streaming, are anticipated to gain more widespread use as the Internet continues to evolve into the common transport platform for all kinds of services. In this article, we examine the specific requirements for key management schemes to be used in ALM-based distribution systems and analyze existing key management approaches with respect to these requirements [1], [2], [3]. Based on the results of this analysis, we design a new key management scheme that combines ideas of the Logical Key Hierarchy (LKH) protocol [4], [5] and the Iolus approach [6]. We compare the resulting scheme to a simple approach that is based on pairwise keys between neighboring nodes without further key-hierarchy based optimization and that serves as a benchmark. Our results of a comparative simulation study clearly indicate the suitability of our scheme for ALM-based live-streaming.

I. INTRODUCTION

Application Layer Multicast (ALM) [7], [8] harnesses the resources of participating end-hosts to create robust and scalable content dissemination systems. The main focus of research so far has been on creating efficient [8] and robust [9], [10], [11] distribution topologies. Security aspects, however, have been largely unattended in ALM research. Especially the establishment of confidentiality among a closed group of nodes is a very important issue, as it is a prerequisite for commercial as well as personal deployment. Content providers, which charge their customers for the service of delivering a multimedia stream, have a strong interest in keeping the transmitted data confidential in order to protect their revenues. Unless the dissemination scheme can offer the same level of protection as traditional unicast channels, content providers are more likely to accept the higher cost of distribution through unicast rather than deploying an overlay dissemination scheme. Confidentiality may also be a key requirement in non-commercial scenarios, too. As participants are sharing potentially sensitive data, they may have a high interest in securing it, restricting access to a closed group of recipients.

To establish a confidential setup in ALM, some key management (KM) scheme has to be introduced in order to agree on common keys to encrypt the content. Key management is an especially difficult task in ALM systems, as they usually do not contain a central or dedicated entity that may take care of key management duties (e.g. serving as a trusted third party, etc.). Considering the distribution of live content yields

additional challenges, as groups in this scenario are typically very short lived and the mean time of participation of nodes is very low [12]. An immediate and central requirement to a key management scheme for an ALM is scalability in the amount of participants. Adding overhead has to be avoided and the resource consumption for necessary overhead should be evenly distributed over all participants. As a direct consequence the introduction of an additional infrastructure component has to be refrained off.

Systems implementing ALM are commonly classified into the categories *mesh-first* or *tree-first* [13]. While mesh-first approaches create a management overlay first and set up the content dissemination topology using this mesh, tree-first approaches create the content dissemination topologies directly and use them for the distribution of management traffic, as well. However, using the ALM for live streaming, the content is actually distributed along a set of spanning trees, each rooted at the source of the stream, that consists of all participating nodes as either inner- or leaf-nodes. Hence, all ALM for live streaming are composed of one or a set of spanning trees, which in consequence can be leveraged for key management purposes.

In this article, we evaluate different types of key management schemes with respect to their adoption in ALM systems for multimedia live-streaming. Based on a qualitative study of the different types of schemes with respect to their suitability in ALM scenarios, we select promising approaches for a closer performance analysis.

Subsequently, we design a decentralized key management approach that is based on the Logical Key Hierarchy (LKH) and the Iolus protocol adapted to a tree-based key distribution topology. In order to reduce overhead, the approach leverages on the spanning tree of the ALM streaming topology. Furthermore, as models of user behaviour [12], [27] have shown the necessity of this prerequisite, our approach is designed to perform under high churn and can handle frequent arrivals or departures of participants of the ALM. A simulation study shows, that this approach performs better than a simple benchmark approach, performing hop-by-hop re-keying based on pairwise secrets between neighbor nodes. However, another lesson learned from both analysis and simulation study is, that in an environment of small groups, which are subject to high churn, the simple and straight forward approach to use pair-wise keys between each pair of neighbors is much more

efficient than the more sophisticated approaches found in the literature.

The rest of the paper is organized as follows. The requirements for a key management scheme to be used in ALM-based live-multimedia distribution are formulated in section II. In section III, we discuss the state-of-the-art in group key management followed by an explanation of our own approach in section IV that is analyzed and evaluated in section V. In section VI finally, we summarize our findings and give a conclusion to our work.

II. REQUIREMENTS FOR A KEY MANAGEMENT SCHEME

In the following, we describe requirements that can be derived from the described scenario and that either relate to performance- or security-related aspects of the key management scheme.

Three major requirements regard the performance of the scheme:

- 1) *Scalability*: The design of the introduced security scheme has to be able to cope with growing, potentially very large numbers ($> 10^6$) of participating nodes.
- 2) *Moderate resource usage*: The key management must not pose high load to the processing, storage or communication resources of any participant. The capabilities of the device of a participant in the group may differ to those of e.g. a central key server. While the latter will usually be a dedicated high-end machine, the former may be a mobile phone or other hand-held device, which is severely limited in its processing and storage resources. For this reason, a participant must not be faced with the task to store a huge amount of keys, or to perform a large amount of expensive computations, e.g. for packet encryption and decryption. In addition, the bandwidth overhead of the KM should be low, so that a re-keying does not lead to a disproportional number of messages. A resource demand independent of the group size is preferable.
- 3) *Ability to cope with churn*: A KM for overlay live streaming needs to be able to cope with a highly dynamic behavior of nodes, especially with high rates of correlated node arrivals and departures, and additionally with nodes that might take part in the service for a very short time only.

The security-related requirements are as follows:

- 1) *Availability*: The key management has to be available without interruption. In consequence, the failure of nodes, especially of a single node, must not lead to a failure of the key management.
- 2) *Minimal trust infrastructure*: The key management scheme should rely on a minimal number of trusted entities only, if possible by leveraging existing infrastructure instead of adding new components.
- 3) *Backward and forward secrecy*: A key requirement, especially for commercial content providers, is to keep both backward- and forward secrecy: A new peer must

not be able to decrypt packets that were sent before it joined the group [14], And a leaving peer must not be able to decrypt any future traffic. These requirements are sometimes also termed “user revocation” or “blacklisting” [15] and the only possibility to keep forward- and backward secrecy is to perform re-keying for the whole group and distributing new keys.

- 4) *Key independence*: New keys should not be encrypted by an old key for distribution, in order to keep forward secrecy, and because an attacker that has managed to gather one key would be able to decrypt the rest of the communication.
- 5) *No 1-affects-n*: Changes in the group should not lead to group-wide re-keying (1-affects-n). Hence, different keys have to be used for different peers: If only a single group key is shared by all participants, the whole group has to be re-keyed, in order to maintain backward and forward secrecy, even if only a single membership change occurs.
- 6) *Controlled access*: An access control mechanisms has to be provided to ensure that only authorized peers are able to join the group. However, access control is out of the scope for this paper.

III. ANALYZED SCHEMES

In large scale content distribution the key management scheme will typically be deployed to distribute one symmetric *Traffic Encryption Key* (TEK), which is used to encrypt the content at the source and decrypting it at the participants, respectively. In order to securely distribute the TEK, a secure context among all participants is created, based on one or several *Key Encryption Keys* (KEK), which again have to be agreed upon using a key management scheme.

Three different strategies to establish a common secure context, and thus to agree on KEK, exist:

- 1) One common KEK for the whole group
- 2) Group-shared KEKs
- 3) KEKs shared between pairs of peers

There are various protocols for KM [1], [2], [3], [16]. Characterizing them with regards to their level of distribution, they can be classified into central, decentralized or contributory schemes [17].

A. Centralized

Centralized approaches are based on a central entity, called group controller (GC), which maintains secure channels to all group members.

1) *GKMP*: The simplest form of centralized group key management is provided by the Group Key Management Protocol (GKMP) [18], [19]. A TEK is distributed via a central GC. In doing so, $\mathcal{O}(n)$ messages and encryptions per change in the group compound are required. In case of a member join, the new TEK is distributed to the former members by using the old key, a characteristic that is in conflict with the requirement of key independence.

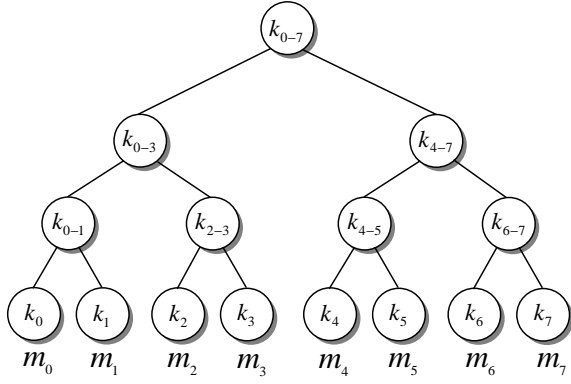


Fig. 1. Logical Key Tree with eight members

2) *LKH*: The Logical Key Hierarchy (LKH) [4], [5] is based on a binary tree of keys (compare figure 1), that is administrated by a central GC. Intermediate keys are created directly by the GC and are encrypted and distributed via pairwise keys that reside in the leaf nodes of the tree ($k_0 - k_7$). The key in the root node of the binary tree k_{0-7} is used as the TEK of the group and all keys in intermediary nodes represent KEKs. The GC is responsible for the key distribution to its members. It transmits to all nodes all keys on the path between itself and their corresponding leaf node. Hence, for every node departure it is inevitable to change all keys known to the leaving member, in order to maintain forward secrecy.

For a secure key distribution, the GC encrypts each key in the tree with the keys of its child nodes and broadcasts them to the group. After a successful authentication of a new member, the GC decides about the joining member's position in the tree and changes all keys on the path between the parent of the new member and the root. Arrivals and departures of members are processed in the same way.

In consequence, a re-keying due to any change in the group causes all children to encrypt the new key, too. For this reason, all keys on the path from its leaf node to the root have to be changed. Regarding the example in figure 1, the case of member m_7 leaving the group lead to the need to change the keys k_{6-7} , k_{4-7} , and k_{0-7} and to replace them with k'_{6-7} , k'_{4-7} and k'_{0-7} . Consequently, the following messages are sent by the GC:

- $\{k'_{6-7}\}_{k_6} \rightarrow m_6$
- $\{k'_{4-7}\}_{k_6} \rightarrow m_6$
- $\{k'_{0-7}\}_{k_6} \rightarrow m_6$
- $\{k'_{4-7}\}_{k_{4-5}} \rightarrow m_4, m_5,$
- $\{k'_{0-7}\}_{k_{4-5}} \rightarrow m_4, m_5,$
- $\{k'_{0-7}\}_{k_{0-3}} \rightarrow m_0, m_1, m_2, m_3$

LKH considerably reduces the number of encryptions compared to a distribution of the TEK via individual channels. Per single join or leave, LKH requires $\mathcal{O}(2 \log n)$ encryptions and the same number of broadcast messages for re-keying in order to change $\mathcal{O}(\log n)$ keys at a group size of n members. If multiple keys are put in one message, the message effort can be reduced to $\mathcal{O}(\log n)$ at the cost of bigger messages.

3) *OFC*: The One-way Function Chain (OFC) [20] relies on pairwise keys between the GC and members and uses a binary key tree, too, which is built up similar to LKH. Upon a member leave all keys on the path from the leaving member to the root node have to be changed. Therefore, the GC transmits a value r to the sibling of the leaving member and applies a hash function f to r to create a hash chain along the path of the leaving member from its leaf node to the root. The GC assigns values from the hash chain according the point the affected path touches the path of other nodes towards the root. In case of m_7 in Figure 1 leaves the group, the following messages are sent by the GC:

- $\{k'_{6-7} = r\}_{k_6} \rightarrow m_6$
- $\{k'_{4-7} = f(r)\}_{k_{4-5}} \rightarrow m_4, m_5$
- $\{k'_{0-7} = f(f(r))\}_{k_{0-3}} \rightarrow m_0, m_1, m_2, m_3$

After that, every member has the TEK, represented by $f(f(r))$, or is able to compute it by hashing the received value one or several times depending on its position in the key tree. A join is handled in the same way. OFC offers a trade-off of computation for communication and requires $\mathcal{O}(\log n)$ broadcast messages per join/leave by applying marginal additional computation requirements. The number of encryptions is $\mathcal{O}(\log n)$.

4) *ELK*: The Efficient Large-Group Key Distribution Protocol (ELK) [21] is similar to OFC and requires $\mathcal{O}(\log n)$ encryptions and $\mathcal{O}(\log n)$ messages per join/leave as well. It is based on a binary key tree and pairwise keys between the GC and all members. In addition, ELK decreases the required message length at the cost of more computation. In case of a member join, intermediate keys are built based on a part of the left and right successor key, respectively. During a member leave, the re-keying message for a certain node contains the missing part for the key calculation. Re-keying messages caused by member joins contain only a checksum of the new key and the members compute the missing bits for the key in brute-force. This leads to a lower communication cost but a considerably higher additional computation effort at the same time, which offends the requirement of only moderate computation effort.

GKMP needs n unicast messages per re-keying and the same number of encryptions. The tree-based approaches like LKH trade-off computation for communication effort. LKH needs only $\mathcal{O}(2 \log n)$ encryptions at the cost of $\mathcal{O}(2 \log n)$ broadcast messages. OFC and ELK perform better in terms of communication and effort for encryptions. Nevertheless, they cause additional costs. In OFC, group controller and group members are burdened with additional computation for hashing. ELK requires significant computation upon a member leave.

Referring to the requirements for group key management schemes in Section II, all centralized approaches suffer from relying on a central entity, which represents a single-point-of-failure and a bottleneck. Scalability and availability can be achieved by additional measures, but is not given by the centralized KM itself. A higher scalability can be accomplished by introducing a virtual server concept and availability can be

achieved by deploying additional redundant group controllers. Centralized KM approaches usually apply symmetric cryptography and for this reason they have only moderate resource requirements. Only ELK needs a considerable computation effort by offering only low communication overhead. Backward and forward secrecy is given in all presented centralized KM approaches. Centralized KM schemes distribute one TEK to the group, posing the need for an immediate re-keying after a member join or member leave, a characteristic commonly known as 1-affects-n. In addition, churn poses a problem to all centralized schemes. Except for GKMP all presented schemes keep the key independence requirement.

B. Decentralized

In decentralized approaches the large group is split up into smaller subgroups and each of them is managed by a subgroup controller.

1) *Iolus*: [6] partitions the overall group into subgroups with one subgroup controller, called Group Security Agents (GSA), respectively. In addition, a Group Security Controller (GSC) represents a central authority and is placed in a top-level subgroup. GSAs are arranged in a tree-based hierarchy with the GSC residing in the top-level group. Neighboring GSA in the tree share pairwise keys. There is not a single global, but one TEK per subgroup. Sending messages from one subgroup to another requires one or several re-encryptions depending on their target destination in the subgroup hierarchy. If a message traverses only one hop, the GSA at the side of the sender takes the message, which is encrypted with the subgroup-TEK, and decrypts it, encrypts it again with the pairwise key of the GSA at the side of the receiver and sends it. The receiving subgroup controller decrypts it and encrypts it again with the subgroup-TEK. Afterwards, it forwards the message to its destination.

2) *Intra-Region Group Key Management Protocol (IGKMP)*: [22] consists of one central Domain Key Distributor (DKD) and several Area Key Distributors (AKD). Each AKD maintains a subgroup. The DKD is responsible for global TEK generation and TEK distribution to the AKDs, which in turn update their subgroups. For a re-keying, the AKDs public key infrastructure can be used as well as secure multicast and logical tree-based algorithms. With the DKD again a central entity is introduced, even though it has not the same tasks as a GC in centralized approaches. As a result of the usage of a global TEK, 1-affects-n and bursty behavior can still be a problem to the KM.

A decentralization of the KM makes it more scalable and the failure of a subgroup controller affects only its corresponding members and not the whole group. Decentralized approaches usually apply symmetric cryptography and therefore they require only moderate computation effort. The communication effort is depending on the specific scheme. The presented decentralized approaches enable backward and forward secrecy and keep the key independence. In Iolus the decentralization mitigates 1-affects-n and improves the ability to cope with churn, because one TEK per subgroup is used and only the affected subgroups have to be re-keyed during a join or leave.

Both approaches still rely on a central entity, namely a GSC in Iolus and the DKD in IGKMP. The second one introduces a globally used TEK, which leads to 1-affects-n and problems with churn.

C. Contributory

In the following, we outline representatives of contributory schemes. Here, each peer has to contribute, via computation and communication, to create a common TEK. The basic idea is not to transmit the final key over the channel according to the underlying Diffie-Hellman key exchange principle. The following schemes are based on that.

1) *TGDH*: The aim of Tree-based Group Diffie-Hellman (TGDH) [23] is to create a common TEK in contributory manner. The basic idea behind is the extension of the two-party Diffie-Hellman key exchange to all members in the group. Therefore, an identical key tree for all members is built. As in centralized tree-based schemes the member nodes reside in the leaf nodes. Sibling members perform a DH key-exchange to create the key which resides in their parent node in the tree. In the initial phase, the members choose a secret s that is located in their leaf node of the tree, respectively, and compute a so-called blinded key based on this value and broadcast it. Thus, a blinded key is the same than a message in the DH key-exchange. A node can distinguish the secret key of a parent node, by its own secret key and the blinded key of its sibling. Therefore, it performs the DH key-exchange all over.

Subsequently, so-called sponsors compute the rest of the blinded keys for the intermediate nodes and broadcast them to the group. Sponsors in consequence are burdened with a higher computational cost than other members. They are chosen depending on the strategy of the implementation. The original publication suggests to use the shallowest rightmost node in the tree. After the exchange of all blinded keys, which are intermediate nodes in the key tree, each member is able to compute the final common key. TGDH needs $\mathcal{O}(\log n)$ exponentiations and $\mathcal{O}(n)$ messages for a join or leave.

2) *STR*: The Skinny Tree protocol (STR) [24] builds up a key tree based on the same mechanisms as TGDH. The main difference is the key tree itself, which is unbalanced. New members are inserted at the leftmost leaf node of the tree. Every intermediate node has two children, one leaf node and another intermediate node. Only the left-most intermediate node has two successive leaf nodes. STR has a constant overhead in exponentiations for inserting a new member, the exclusion requires $\mathcal{O}(n)$. The message overhead per join/leave is $\mathcal{O}(n)$.

3) *FDLKH*: The Fully Decentralized Key Management Scheme on Logical Key Hierarchy (FDLKH) [16] is a derivative of LKH without a central Group Controller. Instead, keys are computed contributory by Diffie-Hellman (DH) key-exchanges. The role of the GC is taken over by so-called captains, which are all affected nodes in the key tree during a join/leave of a member node. During a join, the first intermediate key is computed by a DH between the new member and its sibling. The key one level above is computed

between joining member and the left-most member in the neighbor subtree and so on. In case of a member join, the new group key is encrypted by the old one, which offends the requirement of key independence. For each member join or leave $\mathcal{O}(\log n)$ DH key-exchanges have to be performed.

Contributory approaches are scalable and keep the availability requirement, because no central entities are required. However, the computation and communication effort is very high compared to centralized and decentralized approaches, since for every change in the group compound expensive exponentiations are caused. Due to the use of a global TEK, 1-affects- n and churn poses a problem, because every insertion and every exclusion of a member leads to a re-keying of the whole group. All observed approaches keep backward and forward secrecy and except FDLKH all of them maintain key independence.

D. Discussion

Even though centralized approaches lack of the known problems, like containing a single-point of failure and lacking scalability, they could be judged to be suitable for ALM. However, some further problems occur:

- There is *no central point* in a peer-to-peer live streaming scenario, except the source of the stream. This one should not take over the burden of a GC for all streaming members.
- The *knowledge of the source* should be restricted to its direct successors for security reasons: it should not be known by all streaming members, as it is a highly valuable target for attacks and could be easily identified otherwise.

At the first glance, contributory approaches appear to be suitable, too, since they follow the idea of peer-to-peer by distributing the effort for key computation and communication to all members. However, there are two major concerns for contributory approaches in the streaming scenario:

- The *knowledge of peers* is rather limited. The idea of one-to-many streaming data dissemination is in contrast to key management data dissemination. Neither ring-structures nor tree-structures for contributory key-management apply well and they would have to be set up explicitly.
- *Cryptographic operations* (Diffie-Hellman or Elliptic-Curve) applied in contributory approaches are quite expensive. Even if the total number of operations is low, their effort is not negligible. This situation even deteriorates as the approaches described above leverage the speed of symmetric ciphers for secure communication channels. In other words, contributory approaches need expensive operations while other approaches communicate via already established secure channels and hence leverage from relatively fast symmetric cryptography.

In an environment without a central authority or GC, a centralized KM is not possible. Contributory approaches require knowledge about the group, leading to significant communication and computation effort for single group members. For this reason, a contributory KM is not applicable to

the ALM scenario, neither. In consequence, only the decentralized KM approaches remain as possible solutions to the key management problem in overlay live streaming systems. They provide a scalable KM and without the demanding requirements of a contributory approach. So for example, Iolus or IGKMP can be applied to the presented scenario. Iolus has the disadvantage of the need for a re-encryption of the packets from one subgroup to the other. Both introduce central authorities, however, they are not burdened by the same load as a GC in centralized approaches. In the following section the mapping of a decentralized KM on the ALM system is given.

IV. DESIGN OF A DECENTRALIZED KEY MANAGEMENT FOR PEER-TO-PEER STREAMING

The only central entity in peer-to-peer live-streaming is the source node, which distributes the stream via multiple spanning trees. In order to limit the overhead created by the KM, this existing infrastructure is leveraged by using one of the spanning trees for the distribution of a globally known TEK. To enable backward and forward secrecy the use of a periodic re-keying is proposed. A new TEK is distributed shortly before the old one gets invalidated, so that there is no additional communication delay in streaming.

To establish a global secure context, the source node generates a TEK and distributes it to the group via one spanning tree. Since a member's knowledge about the group is restricted to its direct successors and predecessors in all stripes, the TEK is distributed in a hop-by-hop fashion in the chosen spanning tree until it reaches the leaf nodes. Based on local knowledge, subgroups are generated, which share common KEKs for secure TEK forwarding. All forwarding nodes, including the source, become subgroup controllers and all nodes, except the source, become members in one of the subgroups. Thus, an intermediate node is member in a subgroup managed by its predecessor and is a subgroup controller itself for all of its successors. Participants residing in the leaf nodes of a spanning tree are subgroup members only.

Basically, this is a combination of Iolus and IGKMP. The source as a central point in streaming represents an IGKMP DKD. All forwarding nodes in the streaming are Iolus subgroup controllers, which maintain subgroups with their direct successors. Figure 2 shows a streaming topology consisting of 13 nodes and the resulting subgroups. The TEK created by the DKD is transmitted from the top of the streaming tree to the leaf nodes. The TEK received from an upper subgroup or from the source is decrypted by the subgroup controller, encrypted with the local subgroup KEK and sent further.

For an efficient key distribution one of the centralized approaches described in Section III-A can be applied to generate a subgroup-wide KEK. Based on pairwise keys generated by the DH key-exchange the deployment of LKH in subgroups is proposed. During a change in the group compound it requires little more messages and encryptions than OFC or ELK. The signalling effort can be neglected in this setting, since it is only marginal compared to the transmitted streaming data. The

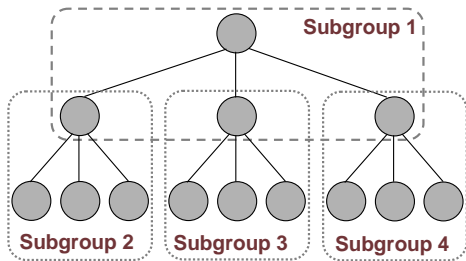


Fig. 2. Decentralized KM, leveraging the streaming topology

number of encryptions for LKH is higher compared to OFC or ELK. However, OFC causes additional computation effort by hashing keys and in ELK significant extra computation is caused by the processing of member leaves. For this reason and because it is the state-of-the-art in tree-based centralized KM, LKH is the approach of choice for a deployment in the subgroups of our KM for ALM.

Our new approach has the following properties:

- *Scalability*: The novel approach is scalable, because of its decentralized manner and the use of local knowledge only.
- *Moderate resource usage*: The decentralized KM for peer-to-peer streaming builds up on symmetric cryptography, except the establishment of pairwise keys, which is done by the DH key exchange. This poses no problem in computation nor storage or communication.
- *Ability to cope with churn*: Joins and leaves of members are processed simultaneously. Furthermore, a re-keying is done only once per interval, allowing multiple insertions and deletions of members at the same time. So, the re-keying of the LKH groups is processed shortly before a new TEK is sent out from the source and not immediately upon a join or leave of a node. In addition, churn is mitigated by the decentralization of the KM leading to small subgroups. In worst-case, at a high churn the whole LKH tree in the subgroup is affected and all keys have to be changed.
- *Availability*: The failure of a subgroup controller leads to a re-organisation of the corresponding spanning tree in the used peer-to-peer streaming approach. Disrupted members reconnect to other nodes and are inserted into other subgroups.
- *Minimal trust infrastructure*: No additional entities for the KM are required, because existing structures are leveraged.
- *Backward and forward secrecy*: A periodic re-keying, instead of a re-keying per join or leave, enables included members to decrypt the group traffic sent via the received TEK before their join and excluded members are able to monitor the group traffic until their TEK gets invalidated. If the re-keying interval is sufficiently small, this poses no problem at all.
- *Key independence*: The key independence is kept, because a new TEK is distributed via the existing KEK

infrastructure in the streaming overlay, instead of the distribution via a former TEK.

- *No 1-affects-n*: Here, the same as for the ability to cope with churn applies. A re-keying is done once per interval and not immediately upon member leave or member join.

For a deeper analysis of the approach a simulation study is presented in the following section.

V. EVALUATION

In order to have a benchmark for our novel approach, it is compared to a simple approach based on pairwise keys between neighbors, which are used for the secure distribution of a TEK. This simple approach allows the exclusion of a member by deleting the corresponding pairwise key before the subsequent re-keying of the TEK.

In contrast, our decentralized LKH-based KM requires an explicit re-keying in every subgroup whenever the group changes, before a new, global TEK distribution can take place. In consequence, a subgroup controller is burdened with additional computation and communication effort. However, this leads to the benefit of saving TEK encryptions during a global re-keying, since only one encryption for all successors at the subgroup controller is required, rather than one encryption per child. The procedure for a member arrival is similar in both approaches: A new member receives a first TEK encrypted with a pair-wise key, to allow an immediate inclusion.

We evaluate and compare our novel KM with respect to computation and communication cost to the simple benchmark approach by integrating both into an ALM simulation framework [25]. In this approach, the forwarding content is split into several *stripes* and a spanning tree is created for each of the stripes. The framework is based on OMNeT++¹ and the Internet Protocol framework INET².

A. Simulation setup

For simulation, the transient build-up phase of a stream was investigated, with an assumed simulation time of 100 seconds and only users joining the stream without leaving ones. In this phase the KM is burdened with the highest load in the course of streaming, since most of the members join at the beginning.

For each set of parameters, $R = 32$ simulation runs are conducted. All results are presented together with their confidence interval, representing a confidence level of 95 percent.

The capacity of the source is set to $C_s = 3$ and a client's capacity to $C_c = 4$. In other words, the server is able to provide the whole stream three times and each client up to four times. Since the stream is divided into stripes, that denotes equal slices of the stream, this capacity can also be used to deliver one stripe C times, i.e. the total capacity in stripes is $T = C \cdot k$, where k denotes the number of stripes. In case of eight stripes, each node may have up to $4 \cdot 8 = 32$ successors in one stripe.

For comparable results for the delay of cryptographic operations we conducted measurements according [26] to gain

¹<http://www.omnetpp.org/>

²<http://www.omnetpp.org/doc/INET/neddoc/index.html>

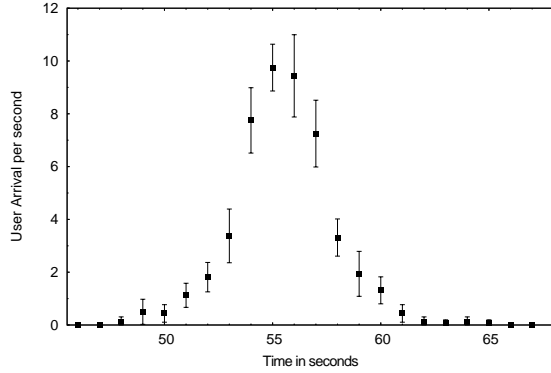


Fig. 3. Resulting join rate in the simulation.

realistic values under the conditions of current hardware. Based on them, for simulation a period of $5 \cdot 10^{-3}$ seconds for a DH-operation with a key length of 1024 bit is assumed and $2 \cdot 10^{-6}$ seconds for a symmetric crypto-operation using AES-256.

The user model is derived from literature [12] and the inter-arrival time follows a Poisson-distribution with a mean of $\lambda = 50$ sec according realistic observations presented in [27]. Figure 3 shows the resulting arrival rate per second in a group of $N = 50$ obtained by 16 simulation runs.

It is assumed that the stream operator requires leaving members to be excluded from the stream as soon as possible. For this reason, the length of the re-keying interval is set to 10 seconds for the TEK, assuming that this is sufficient for a strict exclusion strategy. Consequently, one second before TEK re-keying a KEK re-keying in the subgroups is performed.

Small groups with $N = 50, 100, 150, 200, 250$ clients and large groups with $N = 500, 750$ clients were simulated. The number of stripes is $k = 1, 2, 4, 8$.

B. Simulation Results

The simulation study is intended to answer the question of the suitability of the LKH-based KM approach to the ALM live-streaming scenario. Therefore in the following, the computational requirements and the communication overhead is analyzed in comparison to a TEK distribution solely based on pairwise keys shared by neighbor nodes in the streaming overlay.

1) *Computational results for build-up-phase:* Computational overhead is caused in both approaches by DH key-exchanges and by the overall AES encryptions for TEK and KEK establishment. Both approaches, our novel LKH-based as well as the simple pairwise key-based approach, have the same effort in establishing secure channels. In addition, both approaches require AES encryptions for secure TEK distribution and the LKH-based approach requires additional encryptions for the build up and maintenance of LKH subgroups.

Figure 4 shows the confidence levels of the DH operations performed at maximum nodes, which are the nodes with the highest load per simulation run, respectively. The course of the

graph depends on the behaviour and the characteristics of the streaming topology. Thus, a great impact on the DH operations lies in the amount of stripes k . The maximum number of possible successors for all stripes is $k \cdot C_s = 3 \cdot k$ for the source and $k \cdot C_c = 4 \cdot k$ for a client. Beyond, the allocation of free places in the different levels of the tree has a great influence. Table I shows the maximum depth of the streaming tree and the maximum number of forwarding nodes for all simulated group sizes depending on the number of stripes.

At eight stripes 50 nodes are placed in $l_{max} = 2$ levels and from 150 to 750 nodes, the tree contains $l_{max} = 3$ levels. So, at 150 nodes, not all free places on the three levels of the streaming tree are allocated. Consequently, the most intermediate nodes do not reach their maximum successor number and for this reason they have less DH operations than a node with the full amount of children. This justifies the heavy increasing at the beginning of the graph, containing the results of four and eight stripes, and the stabilization of four and eight stripes at around 150 and 500, respectively. At this points, most of the intermediate nodes have reached their maximum number of successors.

The effort for both source and clients rises linearly with increasing number of stripes. It is further independent on the group size unless the group size is smaller than the maximum number of successors a forwarding client is able to have. The number of forwarders at the same N decreases with more stripes, since the remaining forwarders have more successors.

After an initial connection attempt of a node, a DH key-exchange is performed and the generated key is used as KEK to send the first TEK. This is common for both approaches. In the simple approach a forwarding node distributes a newly received TEK by encrypting it individually per successor. So, one AES encryption per successor is required.

In our LKH-based approach only one AES encryption is needed for forwarding the TEK, since all successors share a common subgroup KEK. Therefore, additional computation effort, in terms of AES encryptions, for the buildup and the maintenance of the key tree is caused. A KEK re-keying in a LKH-based subgroup of n members due to a single change in the group compound requires $\mathcal{O}(\log n)$ keys to be changed

Group size N	Number of stripes k			
	1	2	4	8
50	3	3	2	2
	16	28	4	4
100	4	3	3	2
	52	28	52	4
150	4	3	3	3
	52	28	52	100
200	4	3	3	3
	52	28	52	100
250	4	4	3	3
	52	196	52	100
500	5	4	3	3
	196	196	52	100
750	5	4	3	3
	196	196	52	100

TABLE I
DEPTH OF STREAMING TREE DEPENDING ON N , k AND $C = 4$.

of the simple approach lies in the large amount of TEK encryptions at forwarding nodes, which are burdened with one encryption per direct successor.

In contrast, LKH decreases this number to one TEK encryption per subgroup. Nevertheless, changes in the group during a TEK re-keying interval require a re-keying of the subgroup KEK and in consequence a high churn and a high re-keying interval will cause high computation and communication cost. In the worst case, all keys of the local LKH trees have to be changed and our LKH-based KM is outperformed by the simple approach.

However, our simulation results are based on a user behavior according to [27], [12], which is characterized by a heavy load in the transient build up phase of streaming. In addition, a strict exclusion strategy is assumed with short re-keying intervals, so that the cost at the subgroup controllers for re-keying is decreased. With regards to the computational effort, the LKH-based KM performs better compared to the simple approach in this context, as it demands only a slightly higher communication effort.

Summarizing, the LKH-based KM seems to be suitable for even highly dynamic scenarios. However, when the load increases to a certain point, LKH will be outperformed by the simple approach. Thus, as both approaches are based on pairwise keys, a dynamic adaptation of the KM according to the current load situation in streaming is suggested.

REFERENCES

- [1] S. Rafaei and D. Hutchison, "A survey of key management for secure group communication," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 309–329, 2003.
- [2] Y. Challal and H. Seba, "Group key management protocols: A novel taxonomy." [Online]. Available: <http://citeseer.ist.psu.edu/challal05group.html>
- [3] X. Zou, B. Ramamurthy, and S. S. Magliveras, *Secure Group Communications Over Data Networks*. Santa Clara, CA, USA: Springer-Verlag TELOS, 2004.
- [4] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8(1):16–30, 2000.
- [5] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architecture," 1999, rFC 2627.
- [6] S. Mitra, "Iolus: a framework for scalable secure multicasting," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, pp. 277–288, 1997.
- [7] P. Rodriguez, E. W. Biersack, and K. W. Ross, "Improving the Latency in the Web: Caching or Multicast?" in *3rd International WWW Caching Workshop*, 1998.
- [8] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456–1471, Oct 2002.
- [9] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments," in *19th ACM Symposium on Operating Systems Principles*, 2003, pp. 298–313.
- [10] T. Strufe, "A peer-to-peer-based approach for the transmission of live multimedia streams (German: Ein Peer-to-Peer-basierter Ansatz für die Live-Übertragung multimedialer Daten)," Ph.D. dissertation, TU Ilmenau, 2007.
- [11] S. Birrer and F. Bustamante, "Magellan: Performance-based, Cooperative Multicast," in *International Workshop on Web Content Caching and Distribution*, 2005, pp. 133 – 143.
- [12] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin, "A Hierarchical Characterization of a Live Streaming Media Workload," in *ACM Internet Measurement Workshop*, 2002, pp. 117 – 130.
- [13] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *ACM Computer Communication Review (SIGCOMM)*, 2002, pp. 205–217.
- [14] Y. Challal and H. Seba, "Group key management protocols: A novel taxonomy," *International Journal of Information Technology*, vol. 2, no. 1, Dec. 2005.
- [15] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and efficient constructions," *IETF*, 1999.
- [16] D. Inoue and M. Kuroda, "FDLKH: fully decentralized key management scheme on logical key hierarchy," *Lecture Notes in Computer Science*, vol. 3089/2004, pp. 339–354, 2004.
- [17] C. Abad, W. Yurcik, and R. Campbell, "A survey and comparison of end-system overlay multicast solutions suitable for network-centric warfare," *International Society for Optical Engineering proceedings series*, vol. 5441, pp. 215–226, 2004.
- [18] H. Harney and C. Muckenhirn, "Group key management protocol (gkmp) specification," RFC 2093 (Experimental), July 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2093.txt>
- [19] H. Harney and C. Muckenhirn, "Group key management protocol (gkmp) architecture," RFC 2094 (Experimental), July 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2094.txt>
- [20] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *INFOCOMM'99*, 1999. [Online]. Available: citeseer.ist.psu.edu/canetti99multicast.html
- [21] A. Perrig, D. Song, and D. Tygar, "Elk, a new protocol for efficient large-group key distribution," 2001. [Online]. Available: citeseer.ist.psu.edu/perrig01elk.html
- [22] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang, "Secure group communications for wireless networks," *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, vol. 1, pp. 113–117 vol.1, 2001.
- [23] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *University of California, Irvine*, 2002.
- [24] Y. Kim, A. Perrig, and Gene Tsudik, "Communication-efficient group key agreement." [Online]. Available: citeseer.ist.psu.edu/kim01communicationefficient.html
- [25] T. Strufe, J. Wildhagen, and G. Schäfer, "Towards the construction of Attack Resistant and Efficient Overlay Streaming Topologies," in *2nd International Workshop on Security and Trust Management*, 2006, pp. 108–118.
- [26] W. Dai, *Crypto++ v5.2.1*, Aug. 2006.
- [27] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," *Carnegie Mellon University*, Oct. 2004.