

Engineering Multicast Group Key Distribution a Review

C. Vidakis and M. Fleury

University of Essex

Department of Electronic Systems Engineering,

Wivenhoe Park, Colchester CO4 3SQ

United Kingdom

Tel. +44 (0)1206 872817

Fax. +44 (0)1206 872900

fleum@essex.ac.uk

Abstract

The complex software structures and procedures to establish multicast group key distribution are analyzed, with resulting performance timings and trial applications reported. Existing solutions for very-large scale group key distribution with a hierarchical virtual tree topology are engineered for commercial applications using a public key infrastructure and open source Java security libraries. To solve a major but understated problem with existing schemes, the cost of initial key distribution, privileged members are proposed. Comparison is made with existing proposals showing how the features of this proposal would impact. The paper contains review material on existing schemes, including the security flaws of some schemes.

1. Introduction

Multicast groups have potential commercial applications [1][2][3] such as stock market data distribution, multi-player games, multi-media streaming including local area Internet Protocol (IP) television, and teleconferences. The widely deployed Internet Group Management Protocol (IGMP) [4][5] allows arbitrary multicast groups to be formed, with members able to join and leave groups at will. In the interests of scalability, senders are not aware of potential receivers [6]. Such open groups are unsuitable for the applications cited unless session security is added, for reasons of commercial viability and/or confidentiality, though other applications such as virtual whiteboards or fault-tolerant computing may benefit from open groups, without the need for session security. In effect, the root problem is not the method of group management but the means of managing application sessions within the group, some of which may need to be secure and others of which may not. Therefore, one proposal emanating from the IETF, for example see [7][8][9], is to overlay closed multicast group sessions by means of cryptographic protection of multicast messages. Unfortunately, as soon as multicast

groups become large, beyond 1,000 members in [10], the logistics of key distribution pose a significant threat to the viability of any scheme. For example, to maintain forward security [11][12], whereby leaving members do not have access to any future group key¹, an approach which scales linearly requires re-keying of a new group key to all current members. This would be the case whether IGMP was used or not. For example, if each group member was initialised with their own Key for Encrypting a group Key, a KEK, then in a centralized scheme a single message could be multicast containing multiple copies of the group key, each copy encrypted with a different member's KEK. For a large group over 1000, and using a simple 56-bit Data Encryption Standard (DES) key [15], the message size would be at least 8 kB and, hence, would be fragmented. In fact, groups very much larger than this are being contemplated with [12] mentioning 100,000, [16] mentioning 1 million members, and [17] mentioning 10 million members for pay television. For such large groups, even modest join/leave rates rapidly cause intolerable queues [18] at key server(s). However, it should be noted that for applications with large numbers of members such as pay-to-view, typically with only one sender, perfect forward (or backward) secrecy is not a requirement, as the broadcast takes place at a fixed time after pre-payments.

A considerable number of multicast group key distribution schemes have been proposed, for surveys refer to [16][19][20][9]. The two multicast group key management research forums² [21][22], assuming centralized key management, have focused on: policy matters – how access to the group is authorized and authenticated, and how an encryption scheme is chosen; key management —when should the group key be changed and how is a new shared group key distributed; and data traffic encryption — the choice of suitable encryption methods. However, some of proposals remain complex, may have weaknesses in authentication, and are apparently unproven in the field. Published results may rely on simulations, *e.g.* for the Kronos [18] using the ns-2 network simulator, or simply algorithmic descriptions, rather than a software prototype such as that for Iolus [23]. A number of security flaws [24][25] for at least one class of group key establishment algorithm have been identified (refer to Section 5.2). Mobile and wireless networks for multicast [26] introduce bandwidth restrictions, which for some applications have implications on key distribution latency, and which in its turn is related to message sizes. Such networks are more likely to have device nodes with reduced

¹ A method has t -forward security [13] if and only if no t colluding evicted multicast group members can read any future communications. For some applications, (say) military ones, t -backward security is also necessary, whereby no t new group members can read any previous communications. Perfect Forward Secrecy (PFS) [13] is a differing and often desirable criterion, whereby compromise of long term keys (key encoding keys) does not lead to compromise of past session keys (group keys), and requires authenticated group key agreement, normally through authenticated Diffie-Helman exchange [14]. Group keys are symmetric, as the computational cost of asymmetric (public/private keys) encryption remains prohibitive.

² The two research groups represent the IETF and IRTF organizations.

processing power, which can potentially delay start-up times. Memory on these devices is also likely to be limited, reducing local key storage space. However, if a centralized scheme is employed then it may be the server's memory size that is critical if the group size becomes very large, as the server may hold at the least all of the group members' public keys.

In respect to scalability, the problem can be broken into two parts: (1) initial group key distribution and (2) group re-keying whenever the membership of the group changes. This paper considers procedures and performance on a group member leaving, as a join does not compromise security in the same way as a leave, and, hence, it is possible to avoid the full consequences of re-keying. The initial group key distribution might be accomplished by means of an existing security infrastructure. Point-to-point secure distribution of the common group key to thousands of members may be required. Unfortunately, calculation of local secrets requires manipulation of large numbers (beyond the normal word length of the machine) and exponentiation, which lead to significant delays in computation (even if two-way parallel computation takes place [27]). Even more problematical is the initial scalable distribution of sender-specific keys. Sender-specific keys are required if data traffic is to be authenticated on a per-sender basis. In regard to the second issue, leave or join latency must be minimized along with forward/backward security. Re-keying must also be synchronized so that old keys and new keys are not in use at the same time.

To enable initial key distribution scalability, this paper proposes a set of privileged multicast group members, in addition to a centralized group key manager. This addition significantly improves the distribution performance, without the need for a permanent second tier or hierarchy of group servers as occurs in Iolus [23] and the Intra-domain Group Key Management Protocol (IGKMP) [9][28], which are used for re-keying and in the Iolus case for data delivery as well. Privileged members are a transitory second tier, which, once initial key distribution has taken place, take no further part in group operations and, hence, any second tier no longer exists. With the addition of 10% of privileged members, using a testbed network, results indicate that the initial group key distribution latency is decreased by up to 76% for tree topology key distribution overlays. This scheme does involve trust in third parties and, therefore, is unsuitable for applications with hard security requirements such as defence strategy simulations or military command and control. In these circumstances, the GKC may be built with very expensive tamper hardware and be located behind a locked door with fingerprint authentication, which would result in the privileged members becoming the next point of weakness. The cost of protecting a GKC in this way for many applications [29] might well be prohibitive in proportion to the threat. Our system is intended for circumstances similar to those of Iolus, IGKMP and others, which do trust in third parties. Scalable

Multicast Key Distribution (SMKD) [30] also entrusts multicast-enabled routers to act as domain key managers, but is integrated into the Core Based Tree (CBT) multicast protocol [31], the original ‘sparse-mode’ protocol (Section 2). CBT avoids the multiplicity of multicast routes present in source-based tree schemes, reducing possible attack points [32]. However, as each packet travels via the CBT, the shortest path is no longer always taken and it is possible for routing loops to occur. In [33], a core hierarchy avoids the problem of loops, with a small reduction in delay over CBT. Region-based multicasting may reduce delay in those applications such as multi-media streaming and multi-player games, though we do not claim this advantage for our scheme. However, we have compared the reported features of a range of well-known group key distribution schemes to the current scheme with privileged members.

In fact, we have engineered a working key distribution system, otherwise prudently selecting from existing mechanisms in a concept-proving exercise. For ease of reference, the proposed scheme is designated Secure Multicast Group Key Distribution (SMGKD). In SMGKD, a virtual hierarchical tree topology is employed, with a root, a group key controller, and with a set of pre-selected privileged members. The main thrust of the paper is to describe a successful implementation with publicly available security software. In regard to authentication of members, the paper again advocates a conservative approach for practical implementation, maintaining that a Public Key Infrastructure (PKI), despite some dissatisfaction with (say) the need for off-line revocation [34], is appropriate for a given application such as multicast security. The Secure Socket Layer (SSL) through the Java Secure Sockets Extension (JSSE) [35] is employed for transport layer communication, whenever point-to-point communication is required. The authentication, key distribution software, and test applications have been constructed by means of the Java programming language, which includes a substantial set of security libraries and extensions for cryptography, authentication, and secure data communication. Finally, an important practical feature addressed by SMGKD is ease of key distribution management, as otherwise a scheme is unlikely to gain widespread adoption. In summary, the two main contributions of this paper are to: (1) consider a scheme for those applications that can tolerate a trusted third parties; and (2) demonstrate on a testbed network practical implementation with publicly available software, namely that based on Java software technology.

The remainder of this paper is organized as follows. Section 2 establishes the context for how multicast currently takes place on the Internet. Section 3 describes the main system components of the SMGKD, including virtual hierarchical trees, and the choice of support security software. Section 4 is a detailed analysis of the system with procedures for group key

management. Section 5 serves as a review of related work and allows the features of SMGKD to be compared with those of other schemes. In Section 6, SMGKD's features are compared according to established criteria in the literature. Section 7 contains timing experiments for a moderately sized group as a feasibility study. Finally, Section 8 draws some conclusions.

2 Multicast Context

Multicast introduces efficiency in data transportation, shifting to the network layer the responsibility of delivering packets to all interested end-stations. There are some one-to-many or many-to-many applications that use replicated-unicast or broadcast in order to send data to the receivers, but these mechanisms consume processing power and/or bandwidth. For example, the Parallel Virtual Machine (PVM) [36] utilizes replicated unicast, on the grounds that multicast is not standardized or enabled across all LANs or WANs. If a router is multicast-enabled on an Ethernet LAN then an IPv4 datagram is multicast to all members who have explicitly joined a group (established with a unique IP Class D group address), by an approximate mapping onto an Ethernet multicast address. (If a router is not multicast enabled on a WAN, tunnelling can be employed to by-pass it.) Groups are initiated and members join a group through the IGMP protocol access to the local router. Membership is receiver initiated. Only one copy of an IP multicast datagram passes across any one link of a network in a multicast transmission.

IGMP is a means of forming groups at the intra-domain level. As no inter-domain multicast is standard, routers discover and contact other multicast-enabled routers through a variety of multicast routing protocols [37] such as Multicast Extensions to OSPF (MOSPF), Protocol Independent Multicast-Dense Mode (PIM-DM), PIM-Sparse Mode (PIM-SM)³, Distance Vector Multicast Routing Protocol (DVMRP), and Multicast Border Gateway Protocol (MBGP). In the 1990s, a tunnel-based infrastructure, Multicast Backbone (Mbone) used DVMRP, but since 1997 DVMRP's deployment has shrunk in favour of MBGP and PIM-SM, leading to the demise of the Mbone. MBGP also suffers from instabilities if routing table duplication takes place.

As outlined in Section 1, the open nature of multicast protocols and of IGMP in particular is a barrier to further development of applications. Practical multicast data distribution today offers limited privacy or authentication. Multicast groups are open: any host can join by sending an IGMP message. Any host can send data to a group, without even being member of that group. Senders are unaware of the identities of the receivers and receivers cannot authenticate senders. Therefore, if an overlay is to be deployed, not only must it protect traffic

³ The designations 'dense' and 'sparse' refer to the dispersal of group membership. Dense-mode protocols use broadcast and prune to establish multicast trees, whereas sparse-mode protocols use a shared tree and known rendezvous points.

confidentiality but also there is an additional need to provide two-way authentication. In current multicast protocols, there are a number of weaknesses in terms of management, and vulnerability to attack.

Though we are aware of these weaknesses these are not the focus of this paper. For example, there is a risk of session collision in IPv4, which IGMPv3 seeks to remedy by source pruning. This risk has been accentuated by the small memory (1-2 KB) reserved for group memory in some routers. Because of its large address space, IPv6 considerably reduces the risk of address collision. Access attacks are also possible, whereby a multicast group is bombarded with useless messages. Some applications are vulnerable to traffic flow analysis is possible if no obfuscation of traffic patterns takes place. Though interesting, these issues are not within the domain of this paper and are simply flagged here as open issues.

As multicast is implemented under TCP/IP protocol suite's User Datagram Protocol (UDP), which provides 'best-effort' delivery, it is inherently unreliable. Some schemes such as Efficient Large-Group key Distribution (ELK) [38] make provision for dropped messages, but in general reliable multicast is achieved by a reliability overlay [18]. Reviews of numerous reliable multicast proposals can be found in [5].

In this paper, it is assumed that:

- A multicast group has a single multicast group address; all hosts from a group will receive the data sent to that address.
- Multicast groups are dynamic: hosts can join and leave a group at any time.
- Whether or not to be member of a group is a local decision.
- A group can have any number of members, shape, size, or geographical extent.
- A host can belong to several multicast groups at the same time.
- The sender does not have to belong to the group to which they send traffic.
- Group membership is receiver initiated.

Though the group key management scheme presented in the paper is semi-centralized, this does not mean that applications are confined to one-to-many multicast such as video-on-demand. In fact, using the group key management structure to distribute data would normally be a mistake as it introduces a performance bottleneck. In [23], the group key management system acts as a service to a multimedia streaming application that already has a built-in encryption facility. Apart from one-to-many applications, there are also many-to-many applications, as proposed for some multi-player games [39], which are unsuited to a

centralized structure. There also exist some applications that employ few-to-few multicast such as multicast chat-rooms.

3. The SMGKD System Components

3.1 Virtual hierarchical tree topology

Figure 1 illustrates an example logical or virtual key distribution topology. A virtual tree hierarchy is the basis of a number of schemes [8][9][40]. In a virtual tree hierarchy, only the root and leaf nodes are physical nodes. A virtual node exists simply as a mechanism to partition key distribution. The partitioning is such that collusion between two or more excluded users, *i.e.* former group members is avoided. Keys only exist at the server and with members at leaves, and not with any virtual nodes. Therefore in Figure 1, the IP addresses indicate on which leaf nodes a virtual key is held. The group key controller (Section 4.1) resides on the server and holds all of the virtual key node KEKs (long-term keys), and the current group (session) key. A privileged member holds all the virtual key node KEKs. Members hold only the keys needed for their purposes. For example, Member A.1 holds the group key, and the keys of virtual nodes AB and A. The degree of the tree (binary, ternary and so on) may be unrestricted. There is no formal requirement in SMGKD for balanced trees. However, a very long tree branch will require many more KEKs. Moreover, in LKH [8][9][40] the algorithm no longer scales with an unbalanced tree. In Kronos [18], a four-degree tree was selected. Other schemes such as have been designed with binary trees in mind. For example, One-Way Function Tree (OFT) [41] forms a new KEK by merging keys generated from the previous KEKs of just two children. Binary trees would appear to be least efficient, but in general little consideration of tree shaping appears in the group key management literature.

Key update transmission latencies and storage requirements are likely to grow logarithmically with the number of group members, and, therefore, go some way in addressing the resource limitations issues for mobile computing mentioned in Section 1. The physical network topology may be as general as a single server with a one-to-many connection to all physical nodes, though target topologies are clearly not restricted to this arrangement. Table 1, after [10] but which is also appropriate for SMGKD, summarizes some re-keying costs for a leave with various trees and group sizes. In Table 1, the degree of the tree, d , is the number of branches at each node, whereas h represents the height of the tree.

3.2 PKI, digital certificates, and secure sockets

A PKI [42] is used in the proposed architecture to provide group member authentication and message integrity. In PKI, a hierarchical trust structure is established. The responsibility of certificate authorities (CAs) within the hierarchy is to issue and authenticate digital certificates. The primary purpose of a digital certificate is to confirm that the public key (of a public/private key pair) contained in a certificate is the public key belonging to the person or entity to whom the certificate was issued. A digital signature algorithm for signing the certificate must be implemented. In the system herein, a standard X.509 V3⁴ Certificate [43] public key certificate is associated with each group member. Each certificate, signed by the trusted group CA, includes the group public key. In this work the RSA asymmetric encryption algorithm is employed, though notice that one of the limitations identified in [45] was that the standard before the 1995 revision appeared to mandate RSA. Three-way authentication allows mutual authentication (of client and server) without the need for synchronized clocks.

The well-known SSL) v.3 was employed to exchange secure information whenever a unicast connection (using TCP) was made. An SSL session is established by the creation of a shared secret on client and server, without network transport of that secret, through the Diffie-Hellman (DH) exchange mechanism. SSL is complex though details are summarized in [46].

3.3 Implementation software

The Java 2 programming language version 1.3 was selected to implement key distribution, especially as the distribution contains a set of security classes [35]. The `Cipher` `Cryptography` class forms the core of the Java Cryptography Extension (JCE) [47], providing the functionality of a cryptographic cipher for encryption and decryption. The `Cipher` `KeyGenerator` class provides the functionality of a (symmetric) key generator in two ways: in an algorithm-independent manner, and in an algorithm-specific manner. The Java Secure Sockets Extension (JSSE) provides SSL facilities to encrypt communicated data and/or authenticate communicating peers. The `keytool` utility is the “User Security tool”. It can be used to create certificate requests and to manage keystores.

Additionally, utilities from the OpenSSL project [48] were availed of. The OpenSSL Project is a collaborative effort to develop an open source toolkit implementing Secure Socket Layer (SSL) v2/v3 and the IETF standardization of SSL, Transport Layer Security (TLS v1) [RFC2246] protocols, as well as a general-purpose cryptography library. In particular, as X.509.v3 are mandated as peer certificates in SSL, the certificate management utility was employed to form certificate replies, and the `x509` utility allowed conversion between

⁴ Version 3 includes additional information [44] to that included in version 2.

different certificate formats, especially from the PEM format to the DER format supported by Java. The main software components from the OpenSSL project are summarized in Table 2.

The SMGKD protocol database (Section 4.1) can be implemented by a convenient relational database such as Microsoft SQL Server or Oracle. Microsoft Access is also suitable if the database and the Group Key Controller are implemented on the same host. To access the database from a Java program, the Java Database Connectivity (JDBC) Application Programming Interface (API) [49] provides cross-DBMS connectivity to many SQL databases (as well as other tabular data sources, such as spreadsheets or flat files).

4. SMGKD System Architecture

4.1 System Components

Figure 1 illustrates the components of the proposed SMGKD within a network testbed setting. Details of the implementation are given further consideration in Section 4.2, while this Section analyzes the role of each of the system components.

The Group Key Controller (GKC) is the group member with authority to perform critical protocol actions:

- authenticate group members;
- initiates the group key;
- form the key distribution messages;
- perform the re-keying; and
- report on the progress of these actions.

The GKC's main function is group key management. The GKC multicasts start-up parameters include:

- the identity of the GKC and the selected privileged members;
- the encryption algorithm in use and the key size;
- the tree degree (whether binary, ternary, and so on); in effect the number of members per virtual tree node;
- the start time and announcement time-intervals; and
- the maximum number of SSL connections that the GKC, and any one privileged member can accept.

There is an argument that the maximum number of SSL connections should not be made global, as some privileged members may be better able to deal with more connections than

others. The number of manageable connections depends on the capability of the processor. Implementations may wish to improve SSL's efficiency at a cost in complexity.

A **Database Management System** (DBMS) manages the protocol information. Typically, a table is created for each of the following categories of information: the session; the tree-structure; the tree levels; the protocol parameters; the members; and application information. Issues of database security are beyond the scope of this paper, though these would obviously be of concern as part of a complete solution.

A **CA** processes user certificate requests and creates corresponding signed certificate replies, as part of group member authentication. The replies, as well as other group member information are stored to the database. A CA was implemented by means of OpenSSL utilities.

Group members are authenticated users who have declared an interest in entering a multicast session. A CA through a certificate uniquely authenticates each member. A member's information stored at the protocol database includes an IP address, a common name, status, and the certificate.

Virtual Key Nodes (VKNs) keys (KEKs) are used to implement group re-keying. Only the GKC can create the VKN keys. These keys are distributed via the GKC and privileged members. Each normal group member receives only its parent(s) VKN keys, *i.e.* keys of VKNs on a direct path from the GKC to the normal group member; whereas privileged members store all VKN keys.

Privileged members help the GKC during the initial group key distribution procedure. The GKC decides the required number of privileged members, based on the network infrastructure and the total number of group members. A member is selected or defined as a privileged member during user authentication. An X.509v3 certificate extension is used to define the member as privileged. The main function of privileged members is initial distribution of the group key and VKN keys through SSL connections to members. Privileged members receive the group key, all the VKN keys, and information about the group tree structure. They are able to accept SSL connections, and might be implemented as an add-on feature at routers. Otherwise, privileged members share the same characteristics as normal members.

A **group token** is created by the GKC. The token contains information that the group members need to ensure a controller is authorized to create a group, together with the

operating parameters of the group. In particular, it contains the group key. The group token consists of the following fields:

- the group and session identities (IDs);
- the group action in the current message (Initialization, Re-keying, Application multicast);
- the rekeying interval, *i.e.* the life span of the group key;
- the token version, *i.e.* an identifier to identify the current token;
- a token signature, an asymmetric signature using the GKC's private key; and
- the encrypted token payload, *i.e.* the group key.

The token version number can also provide some protection against replay attacks, and, hence, should certainly be made incremental. Including a time in the token version number is helpful in that respect.

4.2 System procedures

4.2.1 Authentication

Authentication of group members is established by means of PKI. PKI incurs a small performance penalty from asymmetric public/private key encryption, but this is only on around 16 to 20 bytes depending on the hash function. On the other hand, PKI has the important advantage that the overhead at the GKC is significantly reduced. In PKI, the group CA is responsible for authentication, producing and managing members' certificates. As a consequence, network bottlenecks at the GKC are reduced when the group CA takes on authentication responsibilities. A secure bootstrapping mechanism is required to enable the GKC to inform the CA of group and privileged members. Another advantage of PKI is that *in principle* any member can become the GKC, as soon as that member has been recognized by the group CA and accepted by the groups' current members. However, we do not advocate a floating GKC, and a possible form of electing a GKC is beyond the scope of this paper.

The detailed authentication procedure steps are as follows:

1. A member uses the Java `keytool` utility to create a public/private key pair, which is subsequently stored in a local key store.
2. A member creates its certificate request.

3. A member sends its certificate request to the CA. The CA receives a ‘Certificate Request packet’ over a normal socket API connection. The packet contains: a certificate request, additional member information, and possibly a request to become a privileged member. We cannot think of a scenario where a member may need to request to become a privileged member and in general such a possibility presents a security risk. Therefore, the GKC or group administrator, in all but exceptional circumstances, should indicate the privileged members.
4. The CA authenticates the member, creates a member certificate reply, and stores relevant information to the protocol database. The CA now uses OpenSSL shell commands to create the certificate reply, and convert it from PEM format to the Java-supported, DER format.
5. The CA sends back the member certificate reply and its (the CA's) own certificate.
6. A member stores the certificate reply in its keystore, and imports the CA's certificate to a `cacerts` file. Both the member's keystore and the private key file are protected by a pass phrase, which is known only to the local user. The member's keystore and private key are stored locally, but the password is not kept on the computer.

4.2.2 Initial group key distribution

The method of initial group key distribution is the main disadvantage of a hierarchical tree topology, which is why our procedure intends to improve existing procedures by means of privileged members. Initial distribution takes place by means of secure point-to-point connections. Security is based on a member's pair-wise keys. In an environment of over (say) 1,000 members, this is an important performance issue, as the GKC has to make point-to-point connections with each member. Furthermore, authentication between the GKC and a member incurs additional overhead, as there is a mutual exchange of both asymmetric and symmetric keys.

The detailed initial key distribution procedure steps are as follows:

1. The group administrator selects the group parameters. Based on these parameters, the GKC produces a group key, and the tree structure with associated VKNs. The tree structure uniquely identifies leaf nodes based on their IP addresses (or an ID assigned by the GKC and distributed by an out-of-band mechanism), thus avoiding leaf assignment clashes.

2. The GKC multicasts to the whole group the 'Initial Privileged Message' authenticated with the GKC's private key. Those members who are not privileged (with no privileged member certificate extension) simply discard this message. A separate privileged members' multicast group, set-up by an out-of-band mechanism, is also possible but, as the group is only used once, this does not appear to be an efficient alternative. This message contains required group parameters such as the GKC's IP address, the encryption algorithm, and the multicast IP address and port number. The GKC accepts an SSL connection by privileged members (and just from privileged members) only for a designated time period.
3. Privileged members verify the 'Initial Privileged Message' and connect to the GKC using a secure SSL connection. Privileged members then receive the tree structure, the group key and all the VKN keys. The tree structure is utilized to identify the corresponding VKNs for any member. With possession of the keys the privileged members are able to initialise members.
4. If all privileged members have connected to the GKC or the initial time period has elapsed then the GKC multicasts the 'Initial Message'. This message includes the same information as the message in step two. In addition, it contains a list of available privileged members, *i.e.* those that have responded to the step two 'Initial Privileged Message'.
5. The GKC and privileged members accept SSL Connections to enable the initial key distribution. Clearly, SSL connections are required so as to guarantee a secure group key distribution.
6. A normal member connects to a privileged member or the GKC. Firstly a member tries to connect to its local privileged member. If the connection fails, due either to the unavailability of that privileged member or because the maximum number of open connections to that privileged member has been exceeded, then the member tries to connect to with the next available privileged member. The GKC is connected to if it is 'nearer' in some sense than any choice of privileged member. For example, if privileged members are identified as routers (Section 4.1), then a member's node will already know its local router (via the Dynamic Host Configuration Protocol (DHCP)). IPv6 provides 'anycast' with a set of addresses that can each be mapped to a set of interfaces. On sending a message to one of these

addresses, an anycast-enabled router directs the message to the nearest interface (through knowledge of the network topology) with this address. In other words, anycast provides a means of identifying a generic ‘privileged member’ address, without the need for a list of privileged members. Once a unicast connection has been established, mutual authentication takes place. Subsequently, the GKC or privileged member sends the group key and the relevant VKN keys. The member then locally stores these keys in the local key store.

An alternative to distributing a list of privileged members is to make use of ‘anycast’, which is a provision of IPv6.

4.2.3 Group re-keying upon a leave

According to security policy, the group re-keying procedure can be activated either:

On-demand to the GKC — From a graphical user interface, the group administrator can suspend members.

Periodically [18] or through ‘Batch Re-keying’ [50] — Either at regular intervals or based on the number of multicast messages issued, the GKC can automatically activate the re-keying process for the whole or for a segment of the group. With pre-knowledge of leaving times, as for multicast TV programs with fixed start and end times, a limited number of keys may be distributed at start time only [17]. However, [17] would need to perform a complete re-key if random joins and leaves were possible.

Based on the number of ‘Leave Messages’ — This is inefficient, and it may be unnecessary to change the group key for each single ‘leave’ in a large group. The re-keying process is activated when the number of ‘Leave Messages’ crosses a pre-set threshold.

Re-keying takes place according to the following steps:

1. A member sends a ‘Leave Message’ to the GKC or is suspended/evicted by the administrator.
2. The GKC, based on the tree structure, identifies the VKN keys that are to be updated and updates the VKN keys and the group key.

3. The GKC multicasts a 'Re-keying Message' that identifies a leaving member's VKNs. Members sharing the same parent VKN, *i.e.* sibling tree leaves, identify the parent VKN that will change and open an SSL connection with the GKC to receive the new VKN keys. If the rate of re-keying is high then there is an argument that privileged members should persist to avoid saturating the GKC. Here we suppose that the rate is moderate and does not jeopardise the GKC's functioning.
4. Once, the VKN keys are updated, the GKC updates the group key by a set of multicast messages, one for each branch of the tree. Each message is encrypted with the stem VKN key for the branch of the key to which the message is aimed. Those leaves without the appropriate key that receive a multicast simply ignore that multicast.

Group Members can identify if their parent VKN will be updated from the 'Initial Message', and subsequently await updated VKN keys. An important advantage of this re-keying process is that re-keying synchronization is not required. However, some implementers may well prefer the option of devising a robust synchronization method, as there is an efficiency penalty in waiting for a new group key.

4.2.3 Group re-keying upon a join

The GKC can employ the same procedure for a join as for a leave. It is also possible to use a one-way function scheme to generate a group key locally rather than centrally create a new group key as in the manner of LKH+ [51], considered in Section 5.1. Provided the same procedure is not employed for leaves then forward/backward security is not apparently compromised. A distributed clock could synchronize re-keying, provided update latencies of about 25 ms can be tolerated. The Network Time Protocol (NTP) [52] includes a facility for authenticated messages.

5. Group Key Distribution Schemes

This section introduces a sample set of schemes, as in Section 6 these and others are compared with SMGKD. This Section is certainly not intended to be comprehensive. In particular, non-inclusion of a paper does *not* imply that it has not contributed to this field. The schemes exist in various forms, some as algorithms, some as organizational frameworks including algorithms, and some as implemented frameworks. Therefore, the contribution of a

scheme is not simply an algorithm but may be in the form of organizational and implementation techniques. Notice that various taxonomies exist. For example, in [16] schemes are categorized by being centralized, non-centralized, and distributed, while in [53] (refer forward to Table 3) a more differentiated categorization is employed. Unfortunately, some schemes are difficult to categorize. For example, Iolus [23] has a centralized organization, but the centralization takes place within sub-groups. Other taxonomies are possible. For example in [12], five categories are distinguished: (1) simple methods that scale linearly (2) number-theoretic methods, for example based on properties of the Chinese Remainder Theorem [54] (3) extended DH methods, for example [55], and (5) hybrid methods that trade-off number-theoretic security for reduced storage. Consequently, this paper is agnostic on how differing schemes are categorized.

5.1 Example schemes

In [12], a number of schemes are characterised as being of the Simple Key Distribution Center (SKDC) type. Examples are military systems such as BLACKER, STU11/BELLFIELD, EKMS, and commercial systems such as X9.17 and Kerberos. When a request is sent to the KDC to set up a new group with its own group key, having checked the clearance status of the potential group members, the group key is communicated to each member in turn through a pairwise exchange. The KEKs to ensure security over the unicast channel are created by a cooperative key generation protocol such as DH or Photuris (Firefly) [27]. This procedure requires each group member to generate the random bits that make up its contribution to the KEK. A peer review process allows mutual authentication of the two parties to the exchange, KDC and group member, typically using the agency of a CA.

The Group Key Management Protocol (GKMP) [56][57] is a framework for group key management (GKM) for multicast security on the Internet. Unlike in SKDC systems, in GKMP trust is not allowed to reside in a third-party KDC, not itself a member of the group. However, GKMP retains key establishment by pairwise exchange. Some features of the GKMP solution in the receiver initiated variant are that a multicast GKC or group key manager is elected by voting; keys have a time limited lifetime; the GKC generates group key packets whereas other servers generate encrypted multicast traffic. However, as a new group key is changed by encryption using an existing group-wide KEK, forward security is not preserved. To avoid compromise, an entirely new group must be created.

The Logical Key Hierarchy (LKH) [9][40][58] is one of a number of algorithms that employ a virtual hierarchical tree topology of the type already introduced in Section 3.1. Compare the

LKH approach with linear re-keying introduced in Section 1. In LKH, each leaf node of a binary tree, corresponding to a physical member, holds the KEKs of those nodes on a direct path from the leaf to the root, corresponding to the GKC. When a node joins to form an n -strong group, having received its own KEK by unicast, the other KEKs it must hold are changed by multicasting a message from the GKC containing the new KEKs ($O(\log_2 n)$ in all for a balanced tree). Each replacement KEK is encoded twice by the KEK of each of its children, resulting in a message dimensioned $2\log_2 n$ in all. A similar algorithm is applied to leaving members to preserve forward security. In [58], a logical key hierarchy is also advocated, though it is concluded that a hybrid system including an Iolus-like physical hierarchy may map on to some network topologies. Star and complete graphs are considered as well as trees. The authors performance test three re-keying strategies (after a tree leave): 1) user-oriented in which all the new keys held are sent separately encrypted on a key per user basis. 2) key-oriented with a similar number of messages to 1) but with a different key per new key. 3) group-oriented in which a single message is broadcast to all group members. The tests indicate that tree degree four is optimal, that group-oriented re-keying though reducing server load increases user (client) load, while the other re-keying strategies reverse this trade-off.

It turns out (from a suggestion by R. Perlman at an IETF meeting and also envisaged in [59]) that for joining members only, it is simply necessary to synchronize re-keying, as new KEKs can be generated by passing the old KEKs through a one-way function, leading to the LKH+ algorithm [51]. A similar idea was already proposed in [60]. The One-way Function Tree (OFT) algorithm [11][12] uses the same topology as LKH but further refines the message size down to $O(\log_2 n)$, *i.e* halved, by generating one set of KEKs, derived from one set of child keys, after passing through a one-way function. The OFT algorithm has also been updated to form OFT+. However, unlike LKH+ in which a hash function is applied to all affected keys, in OFT+ a hash function is only applied to the group key

Underpinning the VersaKey (Versatile Group Key Management) Framework [51] is the LKH+ algorithm, which is extended by using a flat table rather than a hierarchical tree, with consequent reduction in the number of keys held by a GKC. VersaKey supports a range of closely related schemes for key management, ranging from tightly centralized to fully distributed schemes. VersaKey even allows switching between these schemes on-the-fly with low overhead. Operations have low complexity, $O(\log_2 n)$, for joins or leaves, thus granting scalability for very large groups. VersaKey presents a novel concurrency-enabling scheme, which was devised for fully distributed key management. The core of the framework consists of three approaches, which have different properties, but rely on the same basic principle. The

approaches organize the space of keys that will eventually be assigned to group members in a unique way, without actually generating keys before they are needed. The three approaches differ in some important aspects as they offer the user of the middleware a choice between:

- centralized or distributed key management
- no or some trust in other participants
- varying degrees of load on the participants, and
- tight control of the group or failsafe distributed operation.

Efficient Large-Group Key Distribution (ELK) [38] also uses a virtual hierarchical key and a re-keying algorithm similar to OFT in that a member node generates keys by a Pseudo-Random Function (PRF). Re-keying takes place periodically, which should be compared to Kronos [18] below. ELK uses ‘hints’ to counter the unreliability of UDP-based multicast. A hint is a partial key, the remainder of which key can be generated by trial and error. As one part of a key cannot be generated in this way security is not compromised.

Iolus [23], for reasons of scalability, uses a hierarchical group structure, each sub-group having its own subgroup key. Compared to LKH and OFT, Iolus’ subgroup structure is physical rather than logical. A Group Security Controller manages the top-level group of sub-Group Security Intermediaries (GSIs). A GSI decrypts and re-encrypts data sent from a parent GSI, reducing encryption time by simply re-encrypting a one-time key with the local subgroup key, which is forwarded with the payload. If a new receiver joins the subgroup, the GSA either creates and multicasts the new subgroup key, encrypted with the old subgroup key, losing forward security or uses the linear scheme of Section 1 (with increased scalability because of the reduced number of members in a subgroup). The GSI also directly communicates (unicasts) the new subgroup key over a secure channel to the new receiver. A linear scheme is also employed for leaves. In Iolus the GSI being both key and data distribution manager, is likely to be a bottleneck.

Cliques [55] is an extension of the DH key agreement protocol from two to multiple parties. In a DH exchange, the symmetric key never enters the network. Message security rests on the difficulty of finding the logarithm of a number in modulo arithmetic. All parties must know the initial parameters, a large prime and a logarithmic base. A clique is, therefore, a peer group in which every group member contributes to the selection of the group key. Hence, group DH methods are necessarily distributed. Some security weaknesses of Clique schemes (Section 5.2) have been reported.

Kronos [18] is a novel approach to scalable group re-keying for secure multicast. If a multicast group is re-keyed on each membership change, as the size of the group increases and/or the rate at which members leave and join the group increases, the frequency of re-keying becomes the primary bottleneck for scalable group re-keying. In contrast, Kronos is based on periodic re-keying which decouples the frequency of re-keying from the size and membership dynamics of the group. Another feature of Kronos is that it can be used in conjunction with a distributed framework for key management such as the Inter-Domain Group Key Management Protocol (IGKMP) [9][28] that uses a single group-wide session key for encrypting communications between members of the group.

5.2 Security weaknesses

Rather more serious than the following comparison in Section 6 is that some schemes appear vulnerable to attack as a result of formal analysis [24], which can be achieved with the aid of a software tool such as CORAL [25]. In [24], a number of the set of protocols from the Clique project [55], which are based on extended DH. Typically a one-time member of a group is able to learn a key when other members of the group form a sub-group without that member. Equally in [25] it is shown that group members in Iolus [23] can accept messages from former members of the group and can multicast messages that a former member can read. In general, published work on secure group multicast protocols concentrate on performance issues and do not present systematic evidence of a protocol's validity.

6. SMGKD Compared to Other Schemes

Table 3, which is an expanded and updated version of a Table appearing in a presentation for [53], and now also including SMGKD, takes a wide view of the relative advantages and disadvantages of some approaches in the literature. In *pair-wise* key distribution, each node is assigned its own KEK. The scheme mentioned in Section 1, avoids multiple messages at a cost in message size. Large messages require fragmentation to fit the network frame size, contributing to distribution latency, as the loss probability is increased. The virtual *hierarchical* tree scheme reduces a multicast rekey message's size to $O(kh - h)$ for a k -ary tree of height h , with storage of $O(h)$. The re-key *broadcast* method used by Secure Lock [54], whereby all nodes receive the same constant size rekeying message which is appended to the encrypted data message.

Table 4 is an expanded version of a Table 1 in [51] which also now takes in SMGKD. The Table compares the features of three generic approaches: static (no-rekeying); centralized;

and distributed; along with two approaches which are a mixture of these three: VersaKey; and now SMGKD. It is apparent that, from the criteria of the Table, SMGKD differs from VersaKey in respect to the need to trust other participants, namely the privileged members. SMGKD does not need to synchronize re-keying but this property is traded off against a high initial delay. These final two points of comparison have been added to Table 1, compared to the Table in [51].

Finally, in Tables 5 and 6, based on those in [12] but with estimates added for SMGKD, compare the costs associated with respectively initial group key distribution and coping with a leaving (or evicted) member, *i.e.* group re-keying costs. Apart from SMGKD, all the schemes compared in the Tables are based on binary hierarchical trees. In the Tables, n is the number of multicast group members, K is the size of the key in bits, and h is the height of a virtual hierarchical tree, with $h = \log n$ for a balanced tree. If a key or partial key is generated, then random bits must also be generated.

Further sets of comparative tables are provided in [16].

7. Performance Experiments

This Section explores the feasibility of practical implementation of SMGKD to support multicast applications by means of the testbed network. Due to the small size of the groups tested these experiments do *not* claim to give anything more than indications of times involved. In the Figure 3, three Cisco 2600 series routers run the OSPF dynamic routing protocol [37], with multicast enabled. PIM-DM and PIM-SM are used to exchange routing information. The members' machines are Pentium II or III with dual 10/100 Mb/s Ethernet network interface cards. The operating systems were either Linux Redhat 7.2 (running the Zebra routing programme) or Windows 2000 Pro. To allow simple scaling experiments, more than one member process could be placed on a host machine.

7.1 Moderate scale timing experiment

An experiment with up to 102 members was set up, with 10% of those members privileged (rounding down to the nearest integer number of members). A degree five virtual hierarchical tree was employed. Hence, there were ten VKNs, with a maximum of ten members per VKN. The maximum number of SSL connections was set to ten and the RSA encryption algorithm [15] was set with a key size of 128 bits.

Table 7 defines a number of variables that contribute to the time overhead for group key initialization. Hence, a working formula for the total time to perform an initial group key distribution is:

$$\text{Total group initialization time} = \text{IP} + \text{IM} + \text{TP} + \text{TM},$$

where

$$\text{TP} = \text{NP} * \text{GP}$$

and

$$\text{TM} = ((\text{N} - \text{NP}) * \text{GM}) / (\text{NP} + 1),$$

Where the '+1' in the denominator of the right-hand side expressions arises because the GKC is added in as another privileged member.

With 102 members the recorded times were:

$$\begin{aligned} \text{IP} + \text{IM} &= 0.7 \text{ s} \\ \text{GP} &= 0.5 \text{ s} \\ \text{GM} &= 0.4 \text{ s}, \end{aligned}$$

resulting in a total initialisation time of 12.3 s. A member experienced a 3.3 s interruption as a result of group key initialization. 13.93 KB were distributed to privileged members, with the message components described in Section 4.2.2. Table 8 shows comparative group initialisation times, with and without the use of privileged members for a range of small groups.

Table 9 defines a number of variables that contribute to the time overhead for group re-keying. The GKC makes SSL connections and multicasts the updated VKN keys in interleaved fashion. Therefore, which time is the longest depends on the group composition and tree topology. A working formula for the total time to perform an initial group key distribution is:

$$\text{Total re-keying time} = \text{GO} + \text{GU} + \text{MC} + \text{MO}$$

With the same tree parameters as previously and with 10% of members leaving, the recorded times were:

$$\begin{aligned} \text{GV} &= 0.20 \text{ s} \\ \text{MC} &= 0.10 \text{ s} \\ \text{MO} &= 0.15 \text{ s} \\ \text{TM} &= 0.30 \text{ s} \end{aligned}$$

This resulted in a total re-keying time of 3.00 s. A member sharing the same tree node as a leaving member experienced on average a 2.10 s interruption, whereas a member on a different tree node experienced on average a 0.50 s interruption. The reduced delay experienced by a member making an SSL connection with the GKC is largely accounted for the ability under SSL to cache session keys, thus expediting the SSL handshake. As Table 10 shows, if a binary tree is employed then the total re-keying times are elongated. In this case, for 10% leaving members, a member sharing the same tree node as a leaving member experienced a 2.70 s interruption, whereas a member on a different tree node experienced a 0.9 s delay.

8. Conclusions

Probably because of an American initiative to support very large-scale closed multicast groups, for the purposes of defence simulation, there has been considerable research activity, once the major obstacle of group key distribution was discovered. Coincidental to this initiative, there has been equal interest in enabling private multicast groups, which may afford privacy to individual groups, though to prevent illegal activity an escrowed scheme is also possible. Innovative group key management schemes have arisen as a result of that research including virtual hierarchical trees. The principle challenge has been to establish extremely large groups. However, there is a spectrum of group sizes and of application types, with a range of security requirements. The most immediately applicable applications are: (1) multicast between moderately sized, non-commercial clubs and societies that nevertheless wish to be able to identify members and free of disruptive messages (2) Local broadcast of primarily video streams with moderate commercial gain.

The paper has demonstrated that with open source software it is possible to engineer a solution for moderately sized groups, based on scalable solutions for large-scale groups. The detailed procedures for establishing those groups are described in the paper. The auxiliary software in terms of PKI structure, key databases, peer-to-peer key exchange is considerable. Ineluctably in work of this nature, a comparison is made with the plethora of alternative proposals resulting from the earlier research initiatives. Though efficient re-keying has been demonstrated, initial key distribution is neglected as a one-off overhead. Certainly this will not be the case for extremely large groups, as even for our moderately sized groups, set-up time became an important consideration. The proposal for the scheme in the paper is to employ privileged members at the initial key distribution stage. Choice of privileged member is by an external channel. A number of schemes already exist for delegated security and it would seem that, other than for hard security applications, some compromise with logistics is

necessary. The Internet already has examples of applications such as search engine farms in which geographical dispersion has been found to be necessary. The performance timings of our implementation were found to be competitive but by no means negligible, even for a moderately sized application. In future work, we would seek to balanced solution for multimedia streaming in which security was included with the needs of QoS, as it would appear that some proposed multicast streaming solutions do not account at the same time for the need for commercial confidentiality.

References

- [1] R. Wittman and M. Zitterbart, *Multicast Communication – Protocols, Programming and Applications*, (Morgan Kaufman, San Francisco, CA) 2002.
- [2] S. Paul, *Multicasting on the Internet and its Applications*, (Kluwer, Norwell, MA) 1999
- [3] C. K. Miller, *Multicast Networking and Applications*, (Addison-Wesley, Reading, MA) 1999
- [4] W. Fenner, Internet Group Management Protocol, Version 2, RFC 2236, 1997
- [5] D. Koshiur, *IP Multicasting*, (Wiley, NewYork) 1998.
- [6] S. Deering, Multicast Routing in a Datagram Network, PhD Thesis, Dept. of Computer Science, Stanford University, 1990
- [7] T. Hardjono, N. Doraswamy, and B. Cain, “A Framework for Group Key Management for Multicast Security”, IETF Internet Draft (work in progress), draft-ietf-ipsec-gkmframework-01.txt, 1999, available from the SmuG archive at <http://www.securemulticast.org/smug-drafts.htm> and from <http://www.networkassociates.com>
- [8] T. Harjono, M. Baugher, H. Harney, “Group Key Management for IP Multicast: Model and Architecture”, 10th International Workshop on Enabling Technologies for Collaborative Enterprises, 223-228, 2001
- [9] T. Hardjono, L. R. Dondeti, *Multicast and Group Security*, (Artech House, Boston, MA) 2003
- [10] D. Wallner, E. Harder, and R. Agee, “Key Management for Multicast Issues and Architectures”, RFC 2627, 1999
- [11] D. McGrew, A. Sherman, “Key Establishment for Large Dynamic Groups Using One Way Function Trees”, IEEE Transactions on Software Engineering, 29(5): 444-457.
- [12] D. Balenson, D. McGrew, A. Sherman, “Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization”, IETF Internet draft (work in progress), draft-irtf-smug-groupkeymgmt-oft-00.txt, 1999, available from the SmuG archive at <http://www.securemulticast.org/smug-drafts.htm> and from <http://www.networkassociates.com>

- [13] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, (CRC Press, Boca Raton, FL) 1997
- [14] W. Diffie, “Authenticated Key Exchange and Secure Interactive Communication”, Worldwide Conf. on Computer and Comms. Security and Protection, SECURICOM ’90, Designs, pp. 300–306, 1990
- [15] B. Schneier, *Applied Cryptography*, 2nd edition, (Wiley, New York) 1996
- [16] S. Rafaeli and D. Hutchison, “A Survey of Key Management for Secure Group Communication”, *ACM Computing Surveys*, 35(3): 309-329, 2003
- [17] B. Briscoe, “MARKS, Zero Side Effect Multicast Key Management using Arbitrarily Revealed Key Sequences, 1st Int. Workshop on Networked Group Security, 1999
- [18] S. Setia, S. Koussih, and S. Jajodia, “Kronos: A Scalable Group Re-Keying Approach for Secure Multicast”, *IEEE Symposium on Security & Privacy*, pp. 215-234, 2000
- [19] D. M. J. Moyer, J. R. Rao, and P. Rohatgi, “A Survey of Security Issues in Multicast Communications”, *IEEE Network Magazine*, 13(6):12-23, 1999
- [20] P. Kruus, “A Survey of Multicast Security Issues and Architectures”, 21st National Information Systems Security Conf., pp. 408-420, 1998
- [21] Secure Multicast Group (SmuG), Group Security (GSEC) Research Group, online at <http://securemulticast.org>
- [22] Multicast Security (MSEC) Research Group, online at <http://securemulticast.org>
- [23] S. Mitra, “Iolus: A Framework for Scalable Secure Multicasting”, *ACM SIGCOMM’97*, 277-288, 1997.
- [24] O. Pereira and J.-J. Quisquater, “Some Attacks upon Authenticated Group Key Management Protocols”, *Journal of Computer Security*, 11(4): 555-580, 2003
- [25] G. Steel and A. Bundy, “Attacking Group key Multicast Key Management Protocols using CORAL”, *IJCAR’04*, 2004
- [26] D. Bruschi and E. Rosti, “Secure Multicast in Wireless Networks of Mobile Hosts, Protocols and Issues”, *ACM-Baltzer MONET journal*, 2000
- [27] C. Kaufman, R. Perlman and M. Speciner, *Network Security: Private Communication in a Public World*, 2nd edition, (Prentice Hall, Upper Saddle River) 2002
- [28] T. Hardjono, B. Cain, and I. Monga, “Intra-Domain Group Key Management Protocol”, IETF Internet Draft (work in progress), [draft-irtf-smug-intragkm-00.txt](http://www.securemulticast.org/smug-drafts.htm), 2000, available from the SmuG archive at <http://www.securemulticast.org/smug-drafts.htm> and from <http://www.networkassociates.com>
- [29] R. Anderson and B. Schneier, “Economics of Security”, *IEEE Security & Privacy*, 3(1): 12-13 (Special issue on the economics of security), 2005.
- [30] T. Ballardie, “Scalable Multicast Key Distribution”, RFC 1949, 1996

- [31] T. Ballardie, “Core Based Trees (CBT) Multicast Routing Architecture”, RFC 2201, 1997
- [32] T. Ballardie and J. Crowcroft, “Multicast-Specific Security Threats and Counter-Measures”, IEEE Symposium on Network and Distributed Systems, 2-16, 1995
- [33] C. Shields and J. J. Garcia-Luna-Acheves, The Ordered Core Based Protocol, INFOCOM, v. 2, pp. 884-891, 1997.
- [34] P. Gutmann, “PKI: Its Not Dead, Just Resting”, IEEE Computer, 35(8), 41-49, 2002
- [35] S. Oaks, *Java Security*, 2nd edition, (O’Reilly, Beijing) 2001.
- [36] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine: A Users Guide and Tutorial for Networked Parallel Computing*, MIT, Cambridge, MA, 1994
- [37] K. C. Almeroth, “The Evolution of Multicast: From the Mbone to Interdomain Multicast to Internet2 Deployment”, IEEE Network, pp. 10-20, Jan/Feb, 2000
- [38] A. Perrig, D. Song, J. D. Tygar, “ELK: a New Protocol for Efficient Large-Group Key Distribution”, IEEE Security and Privacy Symposium, 247-262, 2001
- [39] J. Smed, T. Kaukoranta, and H. Hakonen, “Networking and Multiplayer Computer Games—The Story So Far”, International Journal of Intelligent Games & Simulation, 2(2):101-110, 2003
- [40] H. Harney and E. Harder. “Logical Key Hierarchy Protocol”, Internet Draft (work in progress), draft-harney-sparta-lkhp-sec-00.txt, Sparta Inc., 1999, available from Network Associates at <http://www.networkassociates.com>
- [41] D. McGrew, A. Sherman, “Key Establishment in Large Dynamic Groups Using One-Way Function Trees”, TIS Labs at Network Associates, TIS Report #0709, 1998
- [42] R. Perlman, “An Overview of PKI Trust Models”, IEEE Network, 13(6): 38-43, 1999
- [43] ISO, “Information Technology --- Open Systems Interconnection – The Directory: Authentication Framework”, ISO/IEC 9594-8 (also ITU-T X.509) 1995
- [44] W. Ford, “Advances in Public-Key Certificate Standards”, ACM SIGSAC Review, pp. 9-15, July, 1995
- [45] C. L’Anson and C. Mitchell, “Security Defects in CCITT Recommendation X.509-The Directory Authentication Framework”, Computer Communications Review, 20(2):30-34, 1990
- [46] W. Stallings, *Network Security Essentials: Applications and Standards*, (Prentice Hall, Upper Saddle River, NJ) 2000
- [47] J. Knudsen, *Java Cryptography*, (O’Reilly, Beijing) 1998
- [48] The OpenSSL project, online at <http://www.openssl.org>
- [49] G. Reese, *Database Programming with JDBC and Java*, 2nd edition, (O’Reilly,

Beijing) 2002

- [50] X. S. Li, Y. R. Yang, M. G. Gouda, S. S. Lam, “Batch Rekeying for Secure Group Communications”, Int. Conf. On the WWW, pp. 525-534, 2001
- [51] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, “The VersaKey Framework: Versatile Group Key Management”, IEEE J. Selected Areas in Communications, Special Issue on Service Enabling Platforms For Networked Multimedia Systems, 17(8): 1614-1631, 1999
- [52] D. L. Mills, “Cryptographic authentication for real-time network protocols”. *AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 45:135-144, 1999
- [53] P. S. Kruus and J. P. Macker, “Techniques and Issues in Multicast Security”, MILCOM’98, pp. 1028-1032, 1998
- [54] G. H. Chiou and W. T. Chen, “Secure Broadcasting Using the Secure Lock”, IEEE Transactions on Software Engineering, 15(8): 929-934, 1989
- [55] G. Atiense, M. Steiner, and G. Tsudik, “New Multiparty Authentication Services and Key Agreement Protocols”, IEEE Journal of Selected Areas in Communications, 18(4): 628-639, 2000
- [56] H. Harney, C. Muckenhirn, and T. Rivers, “Group Key Management Protocol Specification”, RFC 2093, 1994
- [57] H. Harney, C. Muckenhirn, and T. Rivers, “Group Key Management Protocol Architecture”, RFC 2094, 1994
- [58] C. K. Wong, M. Gouda, and S. S. Lam, “Secure Group Communication using Key Graphs”, IEEE/ACM Transactions on Networking, 8(1):16-30, 2000
- [59] V. Viswanathan, “Unconditional Secure Dynamic Conference Key Distribution”, MSc Thesis University of Wisconsin-Milwaukee, 1996
- [60] R. C. Merkle, “Secrecy, Authentication, and Public-Key Cryptosystems”, Tech. Report No. 1979-1, Info. Systems Lab., Stanford Univ., 1979
- [61] R. Oppliger and A. Albanese, “Distributed Registration and Key Distribution (DiRK)”, 12th Int. Conf. on Information Security (IFIP SEC’96), 199-208, 1996
- [62] G. Caronni, H. Lubich, A. Aziz, T. Markson, and R. Skreta, “SKIP: Securing the Internet”, IEEE Fifth Workshop on Enabling Technologies (WET ICE), 1996
- [63] T. Matsumoto and H. Imai, “On the Key Predistribution Scheme – A Practical Approach to the Key Distribution Problem”, Advances in Cryptology, Proc. Of CRYPTO ’87, pp. 185-103, 1987
- [64] M. Burmester and Y. G. Desmedt, “Efficient and Secure Conference-Key Distribution”, Security Protocols Workshop, pp. 119-129, 1996

[65] A. Fiat and M. Naor, “Broadcast Encryption”, CRYPTO’93, pp. 480-491, LNCS no. 773, 1993

No. of members	Degree (d)	Storage (h+1)	No. of msgs. with 1 key per message (dh-1)	No. of mgs. with multiple keys per message (d-1)h
16	2	5	7	4
2048	2	12	21	11
2187	3	8	20	14
131072	2	18	33	17
177147	3	12	32	22

Table 1: Removal of a member in SMGKD: Overhead messages, storage, and tree organization for varying numbers of multicast group members, after [10].

Component	Description
Ca.exe	Certificate management utility used to create certificate replies (PKCS#10) to certificate requests (PKCS#7)
X509.exe	Multi-purpose command – displays certificate information, edits trust settings, sign certificate request. Converts certificate PEM format to the Java supported DER format.
Ca.cer	The certificate authority’s certificate.
Ca.key	The certificate authority’s private key.
Openssl.conf	The configuration file to set the certificate authority’s policies.

Table 2: Software components for X.509 certification in the OpenSSL package.

Topology and examples	Advantages	Disadvantages
<i>Pairwise</i> GKMP[56][57]	Simple, understandable approach.	Not scalable to large groups. Not suited to providing perfect forward secrecy.
<i>Hierarchical</i> OFT [41], SMKD [30], ISAKMP [27]	Scales logarithmically because of hierarchical design.	Changes to group membership require the group key to change. Addressing required for key messages. Inefficient initial group key distribution.
<i>Broadcast</i> Secure Lock [54]	Anonymous re-keying (prevents traffic analysis). Constant time re-keying.	Large processing overhead, which may not scale.
<i>Distributed</i> DiRK [61], Cliques [55]	Robust: Any active participant can distribute key material.	Trust is distributed. Membership lists must be synchronized.
<i>Subgroup</i> Iolus [23], Hardjono [7][9]	Membership changes only affect subgroup level.	Topology is not inherently robust.
<i>Mixed</i> VersaKey[51], ELK [38], GSAKMP [32], SMGKD	Implemented using a combination of features of existing topologies.	Difficult to implement.

Table 3: Choice of distribution topology in contemporary key distribution schemes, expanded version of a table in [53].

Property	Static Examples: GKMP[56][57], ISAKMP[27], SMKD[30], SKIP[62]	Centralized Examples: Iolus[23], Secure Lock [54], Key Pre-Distribution [63], Spanning Tree[64], Fiat- Naor [65]	Distributed Example: Cliques [55] DiRK [61],	VersaKey [51]	SMGKD
Group-wide key	yes	Iolus: no; others: yes	yes	yes	yes
Dynamic join and leave handled	no	yes	yes	yes	yes
Scalability	no	Iolus: yes; others: no	yes	yes	yes
Perfect forward security	no	no	no	yes	yes
Centralized entity required	yes	yes	variable	variable	variable
Trust in third party required	SMKD: yes; others: no	Iolus: yes; others: no	no	no	yes
Trust in other participants	no	no	yes	no	yes
Memory with each entity required	small	Pre-distribution: huge; others: small	Small	small	minimal
High delay in key distribution	no	Spanning tree: yes; others: no	Initial yes	no	no
High initial delay	no	no	yes	no	yes

Table 4: Comparison of various group key distribution schemes, extended version of a table in [51].

Resource measure	SKDC	LKH	LKH+	OFT	OFT+	SMGKD
Total delay	n	2n	1	3n	3n	$\ll n$
No. of bits broadcast	nK	2nK+h	2nK+h	2nK+h	2nK+h	0
No. of bits unicast	0	0	0	0	0	nK
Manager computation	n	2n	2n	3n	3n	$\ll n$
Max. member computation	1	h	h	2h	2h	1
No. of random bits generated	K	2hK	2nK	nK	nK	K

Table 5: Comparison of the costs associated with initial group key distribution, after [12].

Resource measure	SKDC	LKH	LKH+	OFT	OFT+	SMGKD
Total delay	n	2n	2h	3n	3h	h
No. of bits broadcast	nK+lg n	2nK+h	2hK+h	2nK+h	hK+h	2hK
No. of bits unicast	0	0	0	0	0	h
Manager computation	n	2n	2h	3h	3h	2h
Max. member computation	1	h	h	2h	2h	h
No. of random bits generated	K	hK	hK	K	K	K

Table 6: Comparison of the costs for a leaving (evicted) member – re-keying, after [12].

Variable	Description
IP	Time to multicast ‘Initial Privilege Message’ (Section 4.2.2)
IM	Time to multicast ‘Initial Message’ (Section 4.2.2)
N	Number of group members
NP	Number of privileged members
GP	GKC connection time to privileged member
GM	Average GKC connection time per member
PM	Privileged member connection time per member
TP	Total GKC connection time to privileged members
TM	Total GKC connection time to members

Table 7: Description of variables in calculating total group initialization time.

Group size:	10	15	33	56	102
Normal members only	7	10	18	30	52
10% privileged members	5	5	7	9	12

Table 8: Total time (s) to perform an initial re-keying under SMGKD for a non-binary tree.

Variable	Description
NV	Number of updated VKNs
GV	Time for GKC to update a VKN key
GO	GKC overhead time = GV * NV
L	Number of leaving members
D	Degree of tree
TM	Average GKC connection time to member for update
GU	Number of members to update = $L*(D-1)*TM$ -assumes leaving members are each on different branches of the tree.
FR	time to read and encrypt VKN file + time to make multicast transmission
MC	Time to arrange multicast = FR*D
MR	Time to read VKN key file
MD	Time to decrypt received VKN key and save to file
MO	Member overhead time = MR+MD

Table 9: Description of variables in calculating total group re-keying time.

Group size:	40	70	100
10% leaving members	3.45	3.90	4.80
20% leaving members	3.80	5.60	7.40

Table 10: Total time (s) to perform group re-keying under SMGKD for a binary tree.

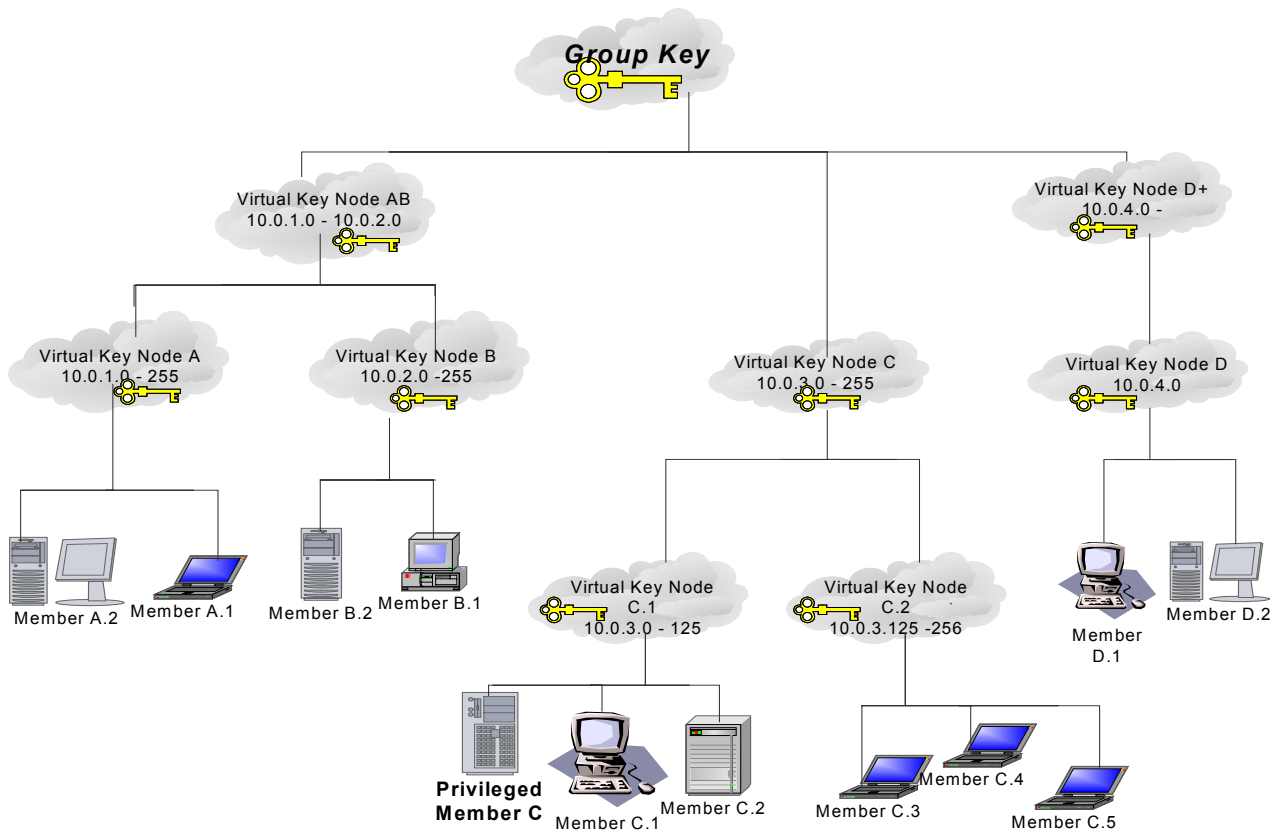


Figure 1: Virtual hierarchical key under SMGKD, including privileged members.

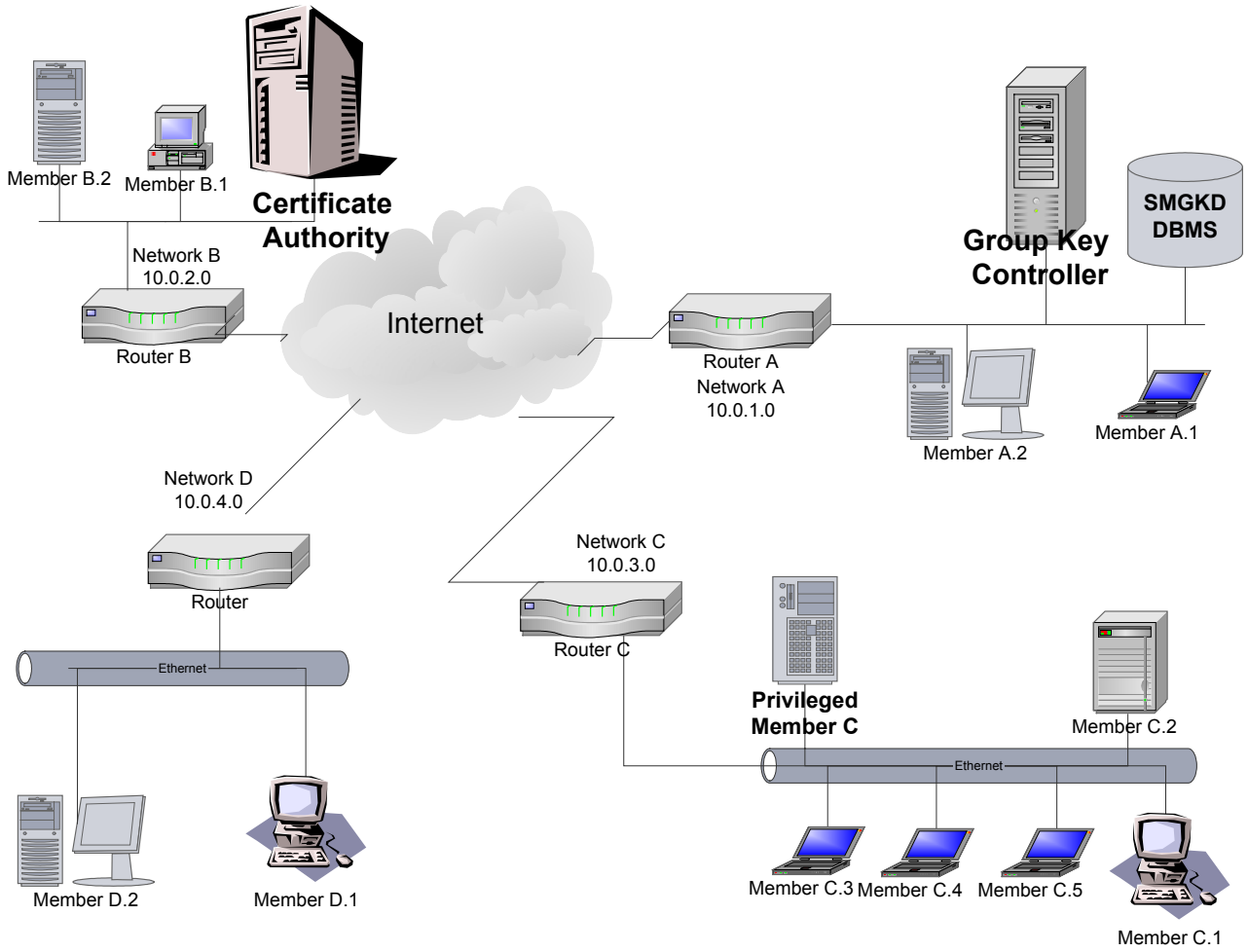


Figure 2: SMGKD system components arranged in an example network setting.

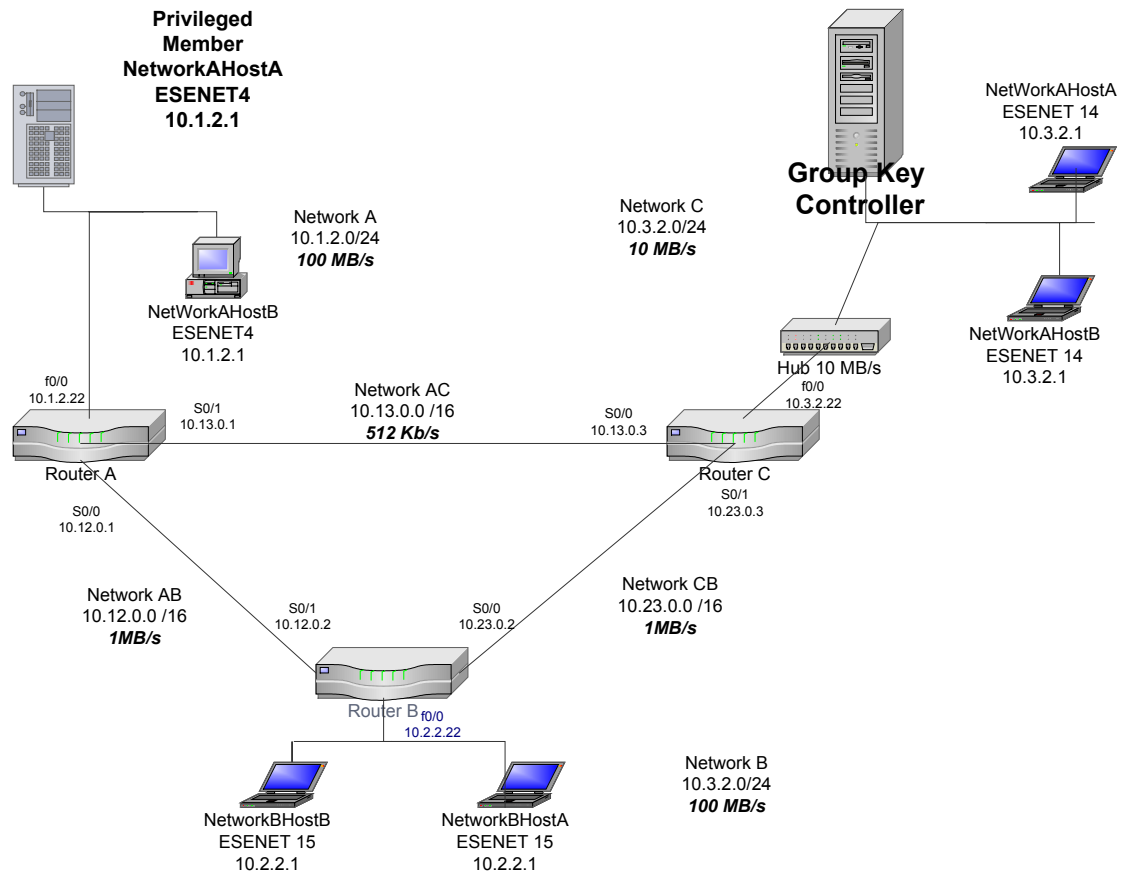


Figure 3: Testbed network for SMGKD.