Key management schemes in multicast environments

Sami Dahlman

University of Tampere
Department of Computer Science
Pro gradu Thesis
November 2001

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Sami Dahlman: Key Management Schemes in Multicast Environments
Pro gradu -tutkielma, 51 pages

Marraskuu 2001

Multicast on IP-protokollateknologia, jonka avulla verkon kapasiteetin käyttöä voidaan optimoida ryhmäsovellusten tarpeisiin. Multicast asettaa turvallisuusteknologiaan liittyvien avainten hallinnalle uusia haasteita. Tässä tutkimuksessa tarkastellaan multicast verkkojen turvallisuusavainten hallintaan tarvittavia mekanismeja. Esittelemme muutamia olemassa olevia avaintenhallintajärjestelmiä ja vertailemme niiden ominaisuuksia.

Avainsanat ja -sanonnat: Multicast, turvallisuus, avainten hallinta

November 2001

Multicasting is an IP protocol technology that can be used to optimise network usage in group communication applications such as videoconferencing. Multicasting sets special requirements for security key management. In this study we take a look at the proposed multicast key management schemes and compare their properties.

Keywords: Multicast, security, key management, scheme

1. Introduction

Group communication services, like news, chat, videoconferencing or online gaming and simulation services, are becoming more popular these days. Also pay-TV and video on-demand type entertainment services are envisioned to become a part of the networked world. The need for security in these environments is very important especially for the service provider and in many applications for the end user, too. The communication occurs most often in Internet type public network. The most important aspects of security in those types of network are that the messaging between group members cannot be eavesdropped and that all the communication between members can be authenticated and integrity checked. These properties are not equally essential in every kind of group application but their need is dependent on the type of trust relationship between communicating parties. For example a pay-TV service provider wants to be sure that that program transmissions are viewed by the valid, paying customers only.

It is very common to make use of a technique called multicasting in the context of group communication over an Internet Protocol (IP) based network. With multicast we are able to save network bandwidth and server capacity by sending some information only once to several recipients at a time. From the security point of view providing an efficient security key management system for possibly a very large multicast group is very challenging as the group members can represent varying degrees of trustworthiness. In this study we will examine various multicast group key management schemes and compare their strengths and weaknesses.

This study is divided into following chapters. In Chapter 2 we introduce IP multicasting and its routing protocols. Then in Chapter 3 we present security concepts needed in multicast key management. In Chapter 4 we present multicast group characteristics and inspect what kind of security issues are related to them. In Chapter 5 we gather a few most well-known key distribution mechanisms and compare their properties. In Chapter 6 we draw conclusions and take a look at the futures of multicast key management.

1.1 Context of the study

To bring the security concepts closer to our world we use an imaginary test bed company whose business relies totally on different kind of multicasting services. The company is called KeyTouch communications and the services it provides can be seen in Figure 1.

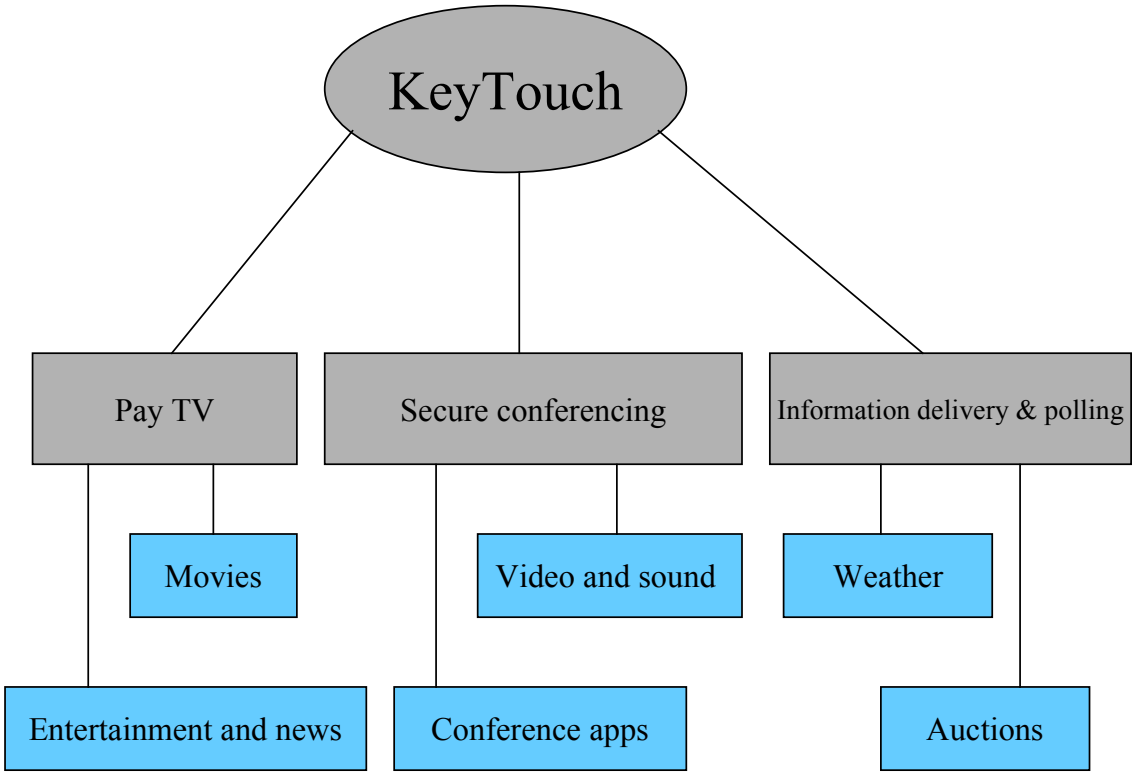


Figure 1. KeyTouch communications services

Keytouch offers services in such a way that every type of multicast communication is used. Pay TV service applies one-to-many communication and secure conferencing applies many-to-many communication whereas information delivery and polling services create both one-to-many and many-to-one communication. All of these services have different bandwidth and delivery delay requirements. Every type of these multicasting communications has also bit different security requirements. [Taxonomy, 1999].

Keytouch is used as an example environment for multicast applications throughout this study so that we really see what kind of implications all the different requirements set to security arrangements.

2. IP Multicasting

Traditional radio transmissions have been with us almost a hundred years and television has entertained us several decades. This idea of broadcasting radio waves through the air (see Figure 2) and cables has truly been used extensively around the world and there is no end to this in the horizon either. Sending information from one point to almost infinite number of receiving apparatus has really revolutionalised society's information delivery mechanism and even our way of life. On the other hand our most natural way of communication - speech - makes use of the same principle. Communication between humans with common language is very straightforward and

transition from human to human and human to group communication is very transparent to us.

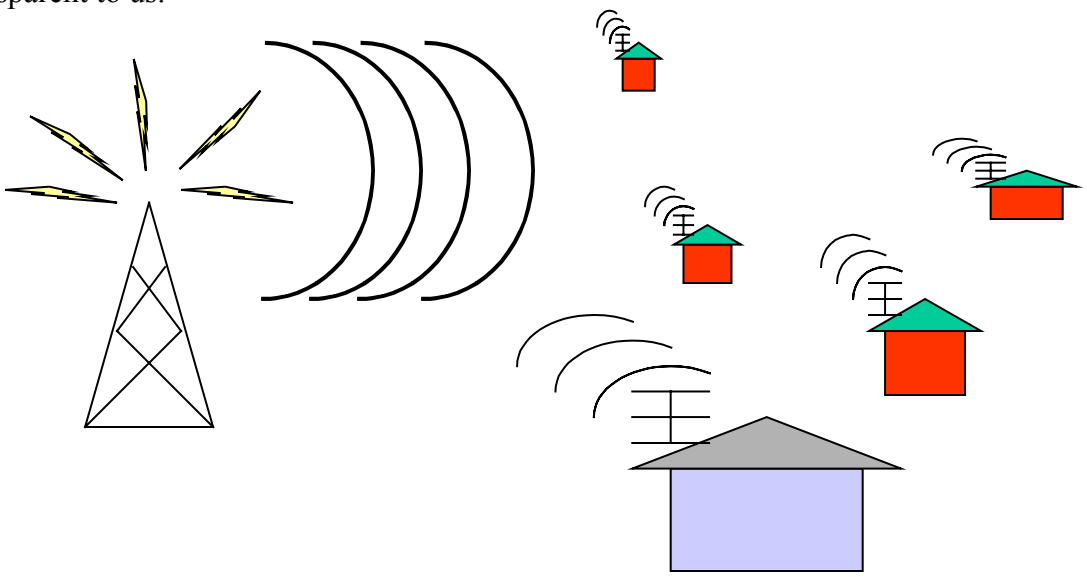


Figure 2. Radio wave broadcasting

When it comes to Internet the traffic has traditionally been only one-to-one or unicast communication, in a sense that the network packets have had only one destination address. There hasn't really been need for other kind of datagrams: people have only connected to remote machines to check their e-mail, transfer files, or view web pages. But this is about to change. At the moment we are witnessing a huge exponential growth of the number of Internet users. These new users are using web services like never before. Static web pages with static pictures and looping sound samples have satisfied people's surfing hunger for a few years now but the services are becoming richer every day. All kinds of streaming media formats, chat communities and gaming services are becoming commonplace. This kind of rich content requires more processor power on the server side and network bandwidth than static content. In practice this means that a popular streaming media server will become a bottleneck, as it can't serve all users adequately when everyone requires separate network packets for same information. This way the capacity of even a clustered server is consumed very quickly.

Another extreme compared to one-to-one unicasting connections is broadcasting. Broadcasting lowers the burden of the server and network tremendously by forming IP network address space based broadcast groups and by sending each datagram only once to this group. This is impractical if we don't want everyone in the same network to receive all the datagrams sent to broadcast group. Things become extremely complicated when we want to route these packets to other networks and still use broadcast. Something else had to be brought up.

In 1985 Steve Deering from University of Stanford developed the idea of multicasting. He came up with a thought that video and audio transmissions could be

sent to a single IP address but still the datagrams would flow to several recipients. This kind of approach meant huge amounts of network bandwidth and server load savings. And already in 1989 Steve Deering had authored RFC1112, the document defining multicast extensions for IP version 4.

As multicasting is an extension to IPv4 standard, it hasn't been extensively used yet because craving for on-demand and group conferencing type services hasn't been very large. Most of the routers in Internet today are simple unicast routers. Newer routers support multicasting so the technology is spreading.

Multicasting requires that all the routers and hosts along the route from sending host to receiving multicast group member support multicasting. With special arrangements we can use IP tunnelling through networks without multicast support to deliver services between multicast islands but this technique is not very commonly used. Nevertheless, this kind of approach is used in experimental multicast network MBONE.

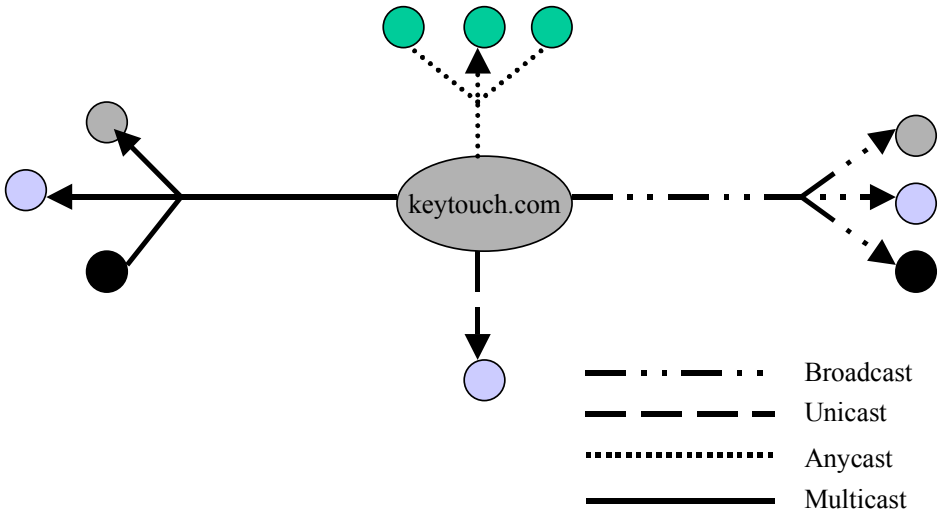


Figure 3. Delivery methods

The implementation of multicasting is mandatory in IP version 6. This means that every router, switch and host supporting IPv6 must also support multicasting. The standardisers have evidently acknowledged that multicast usage will become more popular in the future.

IP version 6 brings also a new type of address called anycast. Anycast addresses can be utilized, for example, when several hosts provide the same service. These hosts share the same anycast address. The user doesn't have to know which host provides the best service as the anycast mechanism decides the most suitable server based on some metrics, for example response time.

2.1 Types of multicasting

There are three types of multicast communication: one-to-many, many-to-many and many-to-one. In every type of communication the idea is the same: The sender directs all datagrams to a single IP address once and the datagrams are delivered to every member of the multicast group.

In **one-to-many** communication there exists only one sender and zero or more hosts that are listening to the senders datagrams. This is natural way for all types of on-demand and file distribution services. Keytouch communications uses one-to-many delivery to telecast movies and all kinds of TV material.

Many-to-many multicast communication occurs usually when we are dealing with group communication. In Keytouch context this means video conferencing and other conferencing services too. More specifically these might include online gaming and online mentoring systems.

Many-to-one type of communication is very useful when we are providing *high availability* resource discovery and data collection services to a large amount users. Keytouch.com provides auctioning service to make use of this kind of communication.

Multicasting is achieved with special routers, which keep track of all the networks within its rooting domain that contain *multicast/host group* members. The routers do not have to keep track of all the members of multicast group. They just need to know the networks towards which they should copy the multicast datagrams. In principle, the sender doesn't have to keep track of all the recipients either. But when we keep in mind the nature of most multicasting services, in practice there has to be some entity on behalf of service provider that registers all the receiving parties. And this is essentially the case from the security point of view too.

2.2 Group Management

Multicasting provides very dynamic framework for managing group members. Members can join and leave groups at any time with the help of Internet group management protocol (IGMP) [RFC 1112, 1989]. A group can contain any number of member hosts. Also, there is no restriction on the number of groups that a host can belong to. IGMP operates only at LAN level between group hosts and closest LAN routers.

In IPv4 IGMP is an extension to Internet Control Message Protocol (ICMP) [RFC792, 1981], which in turn is part of IP protocol. In IPv6 IGMP is an integral part of ICMPv6 [RFC 1885, 1995], as all hosts have to support multicasting [IBM TCP/IP RedBook, 1998].

IGMP is an asymmetric protocol meaning that member hosts are the only ones sending IGMP messages. They communicate with the closest multicast router with join

or leave group messages. The task of the router is then to forward the host's will upstream towards the multicast source or rendezvous point using multicast routing protocols. [RFC792, 1981]

IGMP in its simplicity doesn't provide any means for authenticating hosts or users when they are requesting or quitting the membership of a group. For services like our imaginative KeyTouch.com the authentication of service users is extremely important. So to make multicast services viable for providers and secure for users we need additional services and protocols. We will examine group management security issues in Chapter 4.

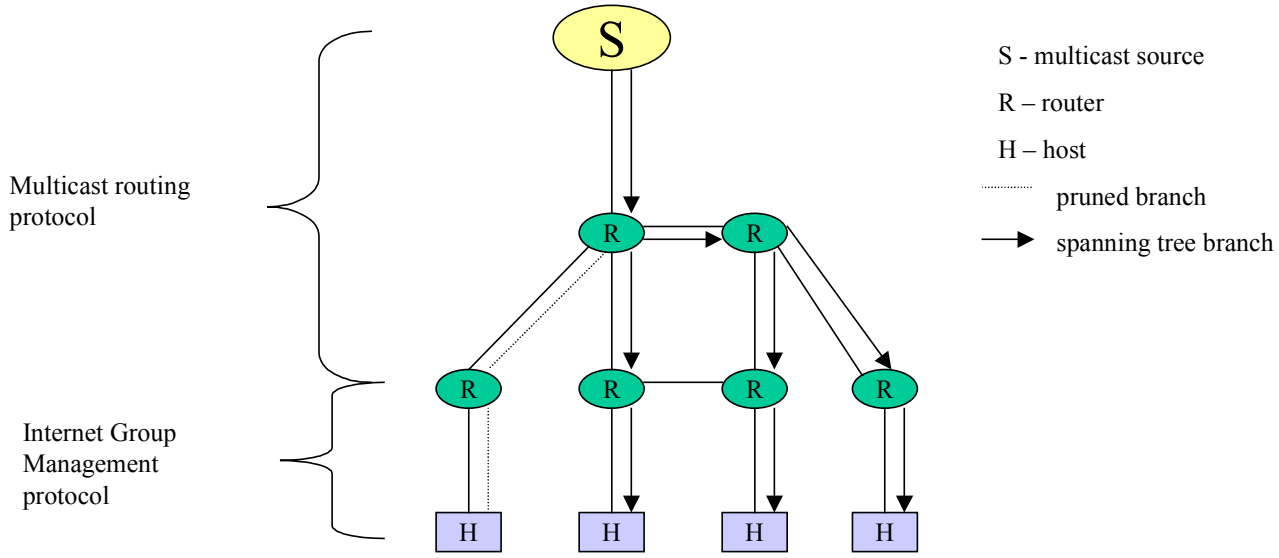


Figure 4. Division of labour between multicast protocols

2.3 Routing

In this chapter we take a look at the available IPv4 routing algorithms and the techniques they are using for packet delivery. This information serves as background knowledge for upcoming multicast security issues.

Most multicast routing protocols use optimal path, loop-free topologies to deliver packets to group members. In fact, routers form spanning trees, which have exactly one connection between each pair of networks (LANs). Routing protocols use different techniques to construct these spanning trees.

Several propositions have aroused considering how to handle multicast routing. In addition to technical differences in propositions, main difference between protocols has been to consider how the group members are expected to be distributed within the network and how the spanning tree is constructed.

One approach is to assume that member hosts are densely distributed throughout the network. These dense area algorithms rely on the facts that every sub-network contains at least one member host and that there is lots of network bandwidth available.

Dense area protocols are usually interior gateway protocols meaning that they are intended to operate inside an organization's network. Dense area protocols use a technique called flooding. This means that routers send multicast packets through every network interface they have as packets move through network and packets are delivered to every densely placed members and the spanning tree is formed.

The most well known dense-mode routing protocols are Distance Vector Multicast Routing Protocol (DVMRP) [RFC 1075, 1988], Multicast Open Shortest Path First (MOSPF) [RFC 1583, 1994] [RFC 1584, 1994] and Protocol Independent Multicast – Dense Mode (PIM-DM) [RFC 2362, 1998].

We say that the member hosts are located in a *sparse area*, if the member hosts are not necessarily located in every subnet and network bandwidth is very limited in some parts of the network. In that type of environment flooding would waste network bandwidth so we need other techniques for forming the spanning tree. Algorithms for sparse areas do not voluntarily send packets to every downstream interface but they need explicit requests for forwarding packets to some interface.

Sparse area algorithms present a new breed of algorithms. Internet community has come to an understanding that flooding algorithms do not scale well to large-scale applications so something else has been brought up to make multicasting really viable. The most well-known sparse area routing algorithms are Core Based Trees (CBT) and Protocol Independent Multicast – Sparse Mode (PIM-SM).

IPv6 brings a number of beneficial features including network level security and quality of service (QoS). Along with these enhancements both unicast and multicast routing become more complex. To accommodate the challenges the research community has started several projects to develop QoS-aware multicast routing protocols. In addition to QoS-aware dense and sparse area protocols some new protocols include adaptive characteristics that enable the building of spanning trees for both densely or sparsely distributed groups.

In the next section we are going to take a brief look at the basic operation and architecture principles of the algorithms in both categories.

2.3.1 Distance Vector Multicast Routing Protocol (DVMRP)

DVMRP is the oldest published multicast routing protocol and its version one dates back to 1988. Since then the protocol has been revised and current version 3 is available as an Internet draft. As stated before it is a dense-mode protocol and is designed to

work within an autonomous system so it does not scale well to large Internet wide applications. [RFC 1075, 1988] It is only able to support spanning trees that are less than 32 levels deep.

Although DVMRP is an old protocol it is very efficient distributed protocol for connection-less multicast packet delivery. In addition to that it has a feature that has helped this protocol to become so popular: DVMRP is able to encapsulate multicast packets into normal unicast packets, which has made it possible to route multicast packets over unicast networks. This has been very useful, as IPv4 does not natively support multicast so most of the routers in the Internet have been without multicast support. The multicast prototype network MBONE has made a lot of use of this feature.

DVMRP is based on the Routing Information Protocol (RIP), which is a unicast routing protocol that has been in use since the early days of Arpanet [RFC 1058, 1988]. Both of these algorithms use the results of the *Bellman-Ford* single-source shortest-path algorithm [Cormen et al., 1990]. DVMRP supports only the number of hops as the routing metric mechanism, which restricts its configurability.

Operation overview

DVRMP uses a “broadcast & prune” technique to build its minimum spanning tree. This means that the multicast routers build a separate spanning tree for each source and destination group by first broadcasting multicast packets to all adjacent routers. They in turn forward the messages by flooding them selectively to all downstream interfaces. This goes on so long as the packets reach all the members and at first this means all the hosts in the internetwork.

The packets are forwarded using a technique called “reverse path forwarding”. When a packet arrives to a router from some interface the router checks the source address from that packet. Then it looks up from its unicast routing table whether the packet came from an interface that provides the shortest path to the source. If that is true then the packet can be forwarded to all appropriate downstream interfaces, otherwise the packet should be discarded. This procedure ensures that if there are routing loops in the network then the duplicate packets are removed from circulation. This way the packets use also the correct minimum spanning tree only to reach their destination.

When a router concludes that it has no members belonging to some multicast group under it, it sends a prune message upstream to indicate that the branch can be removed from the spanning tree. The pruning has a certain timeout, so after the timeout has expired the pruned branch is added back to the spanning tree. If new member hosts from the pruned branch join the group with IGMP join message or downstream routers with dvrmp, the router is able to send a pruning cancellation message upstream so that

the branch can be added back to spanning tree before the timeout expires. [dvrmp-v3, 2000]

With the regular spanning tree-pruning discontinuations DVRMP generates a lot of broadcast like traffic. DVMRP includes also other types of messaging between routers for distributing topology information and packet delivery optimization information. In addition to this routers have to store large amounts of routing information. These protocol level traffic and information storage requirements altogether lead to performance problems when DVRMP is applied in larger networks.

2.3.2 Multicast Open Shortest Path First (MOSPF)

Multicast Open Shortest Path First (MOSPF) is also a dense mode routing protocol, which relies on the services provided by the open shortest path first (OSPF) unicast routing protocol. OSPF protocol uses link-state information to calculate its routes through network. MOSPF is not actually a separate protocol but an extension to OSPF defined in RFC 1583[RFC 1583, 1994]. MOSPF routes packages along least cost paths where the cost is expressed as the link-state metric. DVMRP is only able to use the number of hops as a metric whereas MOSPF can handle other kind of network performance information as a metric; for example load balancing and quality of service metrics are supported. [RFC 1584, 1994]

Operation overview

MOSPF is designated to be used within one routing domain. Each router has an identical, up-to-date routing topology of the whole autonomous system. Each router notifies other routers of any route or topology changes with *link state advertisements*. Routers flood the link state advertisements to all neighbor multicast capable routers. Routers compute the minimum spanning trees using Dijkstra's algorithm [Cormen et al., 1990] and the spanning tree has to be calculated separately for each source and destination group pair. Routers keep also list of multicast group memberships within router's area. In a nutshell the link-state protocols such as MOSPF operation can be divided into three main phases:

1. Gather information about the immediate neighbor routers,
2. Gather information about all the routers in the autonomous system,
3. Calculate the minimum cost spanning trees for all source and destination pairs.

MOSPF suffers of the same performance problems as DVRMP so it does not scale well to large networks. The periodic topology database floodings cause traffic peaks to network and bring the performance down. And in overall the control level overhead is way too high to make MOSPF more popular.

2.3.3 Protocol Independent Multicast

Protocol Independent Multicast (PIM) is one of the newer generations of standardized multicast routing protocols. PIM is able to work in two separate modes: dense-mode (DM) and sparse-mode (SM). Actually these modes are two totally separate routing protocols that do not even share any control messages, but they share the common philosophy of independence of the underlying unicast routing protocol.

Dense-mode

As the name suggests the dense-mode protocol is designed to operate in internetworks where multicast group members are densely located in each subnetwork. In these kinds of environments it is sensible to use flooding techniques similar to DVRMP to deliver the packets in network. So the packets are first sent to all downstream interfaces and then the downstream routers send back prune–messages if there are no members belonging to that group. The pruning is a bit problematic when several routers share a so-called multi-access LAN; this means that the routers share the same physical wire.

This causes problems when one router sends prune message upstream and other routers have still group members in their downstream networks. This situation is tackled with a pruning delay so that when the other routers hear the prune-message they still have to cancel the prune operation. [Kosiur, 1998]

Sparse-mode

PIM-SM does not use the “flood and prune” method. PIM-SM is designed to operate in networks where group members are sparsely distributed and that network bandwidth is not necessarily widely available. PIM-SM does not automatically flood multicast messages to every network but it waits for join requests from downstream routers and after that it starts sending packets to the interface where the request arrived. This way unnecessary packet bursts caused by flooding are absent. From security point of view it is always better not to distribute information to the parts of network where it is not needed.

PIM-SM uses a centralized information distribution and gathering point called rendezvous point. Multicast service providers can register to rendezvous point to market their service and then again multicast audience can look up from rendezvous point what kind of services are available.

Operation overview

The joining procedure from receiver's perspective is similar to dense area protocols. The receiver host uses IGMP join message to notify its directly attached router about the desire to receive some multicast group's packets. The router then sends an explicit PIM-join message towards rendezvous point. Sparse-mode routers have to keep state information about the currently active rendezvous points so that they can send the join request to the correct router. In case there are several routers along a multi-access wire the router with the highest IP-address is selected as the designated router for the LAN. Designated router is the one that forwards the PIM-join requests towards rendezvous point. Whenever the designated router receives an IGMP message, it checks whether the message is destined to a rendezvous point by checking the group address. When the unicast PIM-join request travels towards rendezvous point all the routers along the way create a forwarding entry for the group in question. And then again, when a receiver wants to leave the group it sends an IGMP leave message to designated router and the router forwards the PIM-prune message towards rendezvous point and possible sources. The routers along the way remove the forwarding entry and the rendezvous point removes the branch from spanning tree assuming that no other receivers share the same branch. [Kosiur, 1998] [RFC 2362, 1998]

When a source wants to send a multicast packet to some group its designated router encapsulates the multicast packet in a PIM-SM-Register message, which is then sent to rendezvous point. The active rendezvous point replies with a PIM-Join message and sends it towards the source's designated router. The routers between rendezvous point and the designated router notice the join message and create a forwarding entry for packets coming from designated router. The designated router has to encapsulate multicast packets into PIM-SM-Register packets until it receives a PIM-Register-Stop message from rendezvous point. After that the multicast packets can be sent without encapsulation. [RFC 2362, 1998]

So far we have assumed that in PIM-SM all multicast packets are circulated via rendezvous point before actual multicasting towards receivers is performed. The packets do not necessarily use the shortest path to receivers. Rendezvous point might also become a bottleneck and a point of failure due to its centralised nature. To overcome these problems PIM-SM provides means for optimising delivery tree and balance rendezvous point's load. The receiver's dedicated router can decide based on some metric to construct a source-based shortest path delivery tree. This can be achieved by sending a PIM-Join message directly to the multicast source after packet delivery through rendezvous point has started. After the shortest-path delivery tree is working the rendezvous point rooted delivery tree can be pruned with PIM-Prune message.

2.3.4 Core Based Trees (CBT)

Core Based Trees (CBT) is a multicast routing protocol that has very similar operation principals to PIM-Sparse Mode. CBT uses also a shared distribution tree for multicast packet delivery and is unicast routing protocol independent. Then again CBT does not provide a mechanism for optimizing delivery tree to shortest-path delivery tree but the packets have to circulate through a core point or a rendezvous point as in PIM-Sparse Mode. Also, as the delivery trees are unidirectional in PIM-SM spanning from source or rendezvous point to receivers, in CBT the routers keep bi-directional state information that enables two-way communication between source and receivers. [Kosiur, 1998] [RFC 2201, 1997]

Whereas the shared distribution tree brings scalability to the architecture, it may also become a bottleneck, as a lot of traffic has to be routed via the rendezvous point to the multicast group. The rendezvous point may also be farther away from some senders than others, which may also cause annoying delay.

3. Security concepts

3.1 History

People have used different means for hiding information from unwanted audience for thousands of years. People have always had secrets and a lot of effort has been put throughout centuries to the development of efficient information hiding techniques. Especially wars have made people to produce many hiding methods for securing strategic messaging [Kerckhoffs, 1883]. Most methods used can be classified to cryptographical and steganographical methods. Shortly, steganography tries to hide the existence of some message whereas cryptography uses transformations on the message so that it cannot be understood. These two methods do not try to solve same security problems. They are like different layers of security.

3.1.1 Steganography

Steganography (Greek: covered writing) uses such methods as invisible inks, character arrangements, open messages, covert channels and spread-spectrum communications. Even indians' smoke signalling can be considered to be a steganographical method. In digital environments steganography can be used to hide additional information into digital information formats. These include among others video and audio streams and pictures.

A current active area of research in steganography is the usage of digital watermarking. Digital watermarks can be used to insert virtually irremovable "fingerprints" into copyrighted digital material. From copyright owner's and also multicast service provider's point of view this is very useful as all the digital streams can be stamped with the receiver's identification for example.

3.1.2 Cryptographical substitution methods

First known usage of cryptography dates back about 4000 years. Around 1900 BC in Egypt a scribed carved hieroglyph symbols mixed with some secret symbols to monuments to store and encode information. Similar substitution systems, where a symbol is replaced with another, were also independently developed among other cultures. For example Mesopotamians and Hebrews have developed substitution systems. [Deavours et al., 1998]

The most famous historical cryptography system is the Caesar cipher formulated by Julius Caesar (100 – 44 BC.). In the Caesar cipher each alphabetical letter is shifted three letters forward in alphabetical order.

Example

The Caesar cipher mapping table:

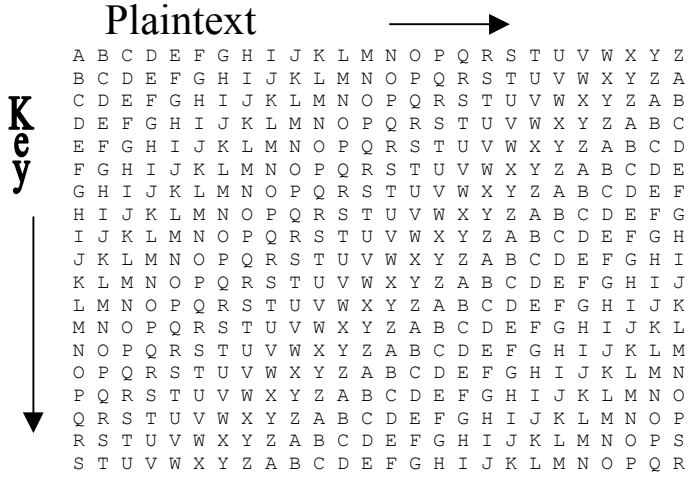
Cleartext	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext	DEFGHIJKLMNOPQRSTUVWXYZABC

Example word:

Cleartext	KEYTOUCH
Ciphertext	NHBWRXFK

In one word case the Caesar cipher may seem very effective, but an experienced cryptanalyst will find out pretty soon that the original Caesar cipher has been used. In general substitutions where a certain alphabetic letter is replaced by another are known as monoalphabetic substitutions. With today's knowledge they are very easy to break but at the time of their invention for example reading ability was not so common so they were considered stronger.

Whereas monoalphabetic systems always produce the same ciphertext letter from a cleartext letter and can be solved in a matter of minutes, polyalphabetic substitutions use more clever techniques to generate different ciphertext letters from a cleartext letter. Polyalphabetic substitutions, as the name suggests, use several different cipher alphabet permutations to produce ciphertext output from cleartext. Many polyalphabetic substitution systems use a two-dimensional array or a tableau as shown in Figure 5.



```
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
Z A B C D E F G H I J K L M N O P Q R S T U V W X Y
```

Figure 5. Polyalphabet tableau

The table rows repeat the normal alphabet from left to right. After each complete row the whole alphabet is shifted one letter to the left. The number of rows is determined by the number of letters in the alphabet. One can encipher messages with polyalphabet tableau by first selecting the cleartext letter to be enciphered from the first row and then a certain row is selected depending on the polyalphabet system to be used. The letter found from the coordinates designated by these two components is the ciphertext letter.

One of the earliest known polyalphabet systems was developed by a German monk Johannes Trittenheim and published in his six book series 'Polygraphia' in 1518. In his system the cleartext column is selected as described above and the row is defined by *n mod x* where *x* is the number of letters in the alphabet and *n* is the sequence number of the cleartext letter.

Example

Here is an example word enciphered with Trittenheim's Polyalphabet substitution method.

Cleartext **KEYTOUCH**

Ciphertext **LGBXTAJP**

Even with this simple Trittenheim's row selection system polyalphabet substitution is far stronger method than monoalphabet substitution. Cryptanalyst cannot make any kind of assumptions of the words based on letter frequencies in cipher text as cleartext letters are mapped to different ciphertext letters each time. All in all, Trittenheim's ground work for polyalphabet systems laid down a base for others to develop it further.

Next logical step was to enhance the row selection method of polyalphabet systems. In year 1563 Italian Giovanni Battista Porta (1535-1615) published a book "De furtivis literarum" where he had documented certain Giovan Batista Belaso's idea of shared secret to be used in row selection. Belaso had already presented the idea in 1553, but Porta finally analysed and documented it indepth and made some improvements.

The idea of shared secret in row selection is to use a keyword that is placed over the cleartext and repeated as necessary as shown below with a keyword 'salakoje':

Keyword: SALAKOJE SAL AKOJESALAKOJ
Cleartext: KEYTOUCH AND MULTICASTING

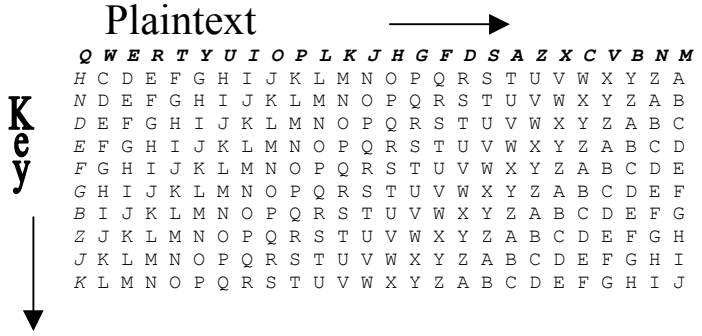
The keyword letter combined with cleartext letter determines the coordinates of the enciphered letter in the polyalphabet tableau. The keyword letter over the cleartext letter under inspection shows directly the row of the ciphertext letter.

Keyword: SALAKOJE
Cleartext: KEYTOUCH
Ciphertext: CEJTYILE

Keywords make polyalphabet substitution a strong ciphering system but not unbreakable. It's very important that the row selection method does not follow any logical pattern. Even when the cryptanalyst knows that polyalphabet substitution has been used it takes a lot of time and effort to find out the key and the cleartext if the key has been selected wisely. The more random the key is the better.

Porta's keyword based row selection was a clear improvement compared to Trittenheim's initial version but it left still room for improvement. An Italian mathematician and physician Girolamo Cardano (1501-1576) thought that the use of only one static key in a message could be improved with an autokeying system. In an autokeying system the cleartext message itself is used as a keyword. Despite the fact that Cardano introduced the concept of autokeying his autokey system had severe design flaws so we do not present it here. The first real usable autokey system was presented by Blaise de Vigenère (1523-1596) in 1585.

Vigenère's autokey system uses the plain text as keying material and it has only one letter as a static keyletter or a primary key. This keyletter and the order of the tableau's side alphabets are the secret pieces of information that has to be known by the sender and the receiver. The idea is to encipher the first letter with the keyletter and after that the first plain text letter acts as the key for the second letter and the second plain text letter is the key for the third letter and so on. Vigenère also presented that mixed alphabets should be used at the sides of the polyalphabet tableau to make the predictability of the usual polyalphabet table a little harder (see Figure 6).



```
L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
C O P Q R S T U V W X Y Z A B C D E F G H I J K L M
O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
A R S T U V W X Y Z A B C D E F G H I J K L M N O P
R S T U V W X Y Z A B C D E F G H I J K L M N O P S
S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
W U V W X Y Z A B C D E F G H I J K L M N O P Q R S
U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
T X Y Z A B C D E F G H I J K L M N O P Q R S T U V
X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
I A B C D E F G H I J K L M N O P Q R S T U V W X Y
```

Figure 6. Viginere's polyalphabet tableau with mixed alphabets at the sides.

Example

We use letter ‘g’ as the keyletter and table X as the ciphering tableau for our example .

Cleartext: KEYTOUCH
Ciphertext: RMYCEUPA

One might think that when we are using only one letter as a keyletter the cryptanalyst can easily check all the alternatives in the alphabet to decipher the message. One recommended way to counteract this is to place nulls at the beginning of the message or possibly encipher the message several times using different keyletters. In any case this ciphering system was very strong at its time.

Polyalphabet substitution systems were considered too complex when they were first introduced so they didn't gain much popularity then. People continued to use simpler monoalphabet substitutions and they were considered efficient enough at their time.

Cipher devices with rotor substitution

One gadget that made polyalphabet cipher a great deal easier was developed by the famous American politician Thomas Jefferson (1743-1826). Jefferson’s device was comprised of 26 discs and a rod. The letters of the alphabet were engraved in some random order on the edge of each disc. The discs were threaded onto the rod in a specified order. The device was used to encrypt cleartext in 26 letter blocks as designated by the number of discs. The cleartext block was encrypted by first spelling out the letters of the cleartext by rotating the discs to the cleartext block. Then another letter line is selected as the ciphertext.

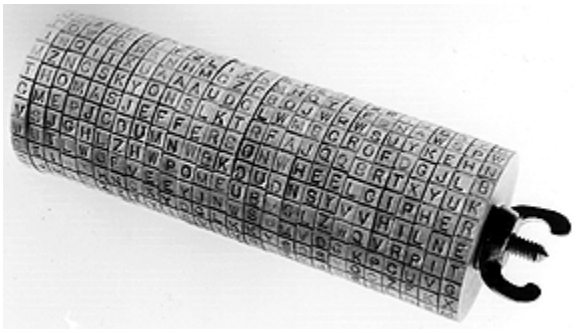


Figure 7. Jefferson wheel cipher

Example

CLEARTEXT "KEYTOUCH PROVIDES MULTICASTING SERVICES"
Encryption of the first cleartext block (26 letters)

We select the line right above the cleartext as the ciphertext to make the example more readable.

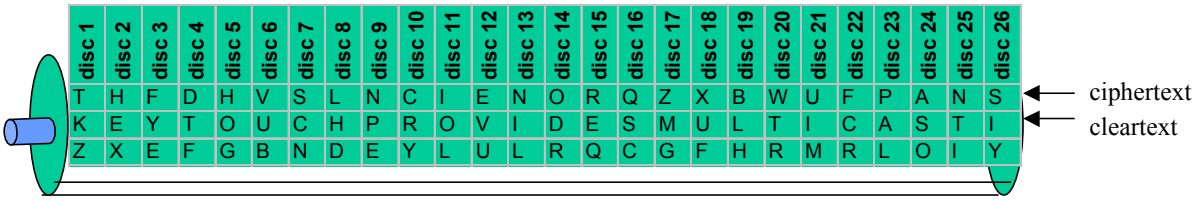


Figure 8. Jefferson wheel cipher

To decrypt the message the receiver has to have exactly an identical Jefferson wheel device and knowledge of the line offset from the cleartext to the ciphertext. If the cleartext message made sense the line offset was not necessarily needed as it could be directly read after the ciphertext had been entered. Jefferson's wheel did not need any additional polyalphabet tableaux or
secret keywords, which made its usage easier than earlier systems. Jefferson's wheel was considered a very strong encryption mechanism at its time. American soldiers used Jefferson wheel cipher for securing communication still in the First World War although soldiers considered fiddling with the small discs as cumbersome.

The modern era of encryption machines was started by the Germans when they developed their own rotor based polyalphabet substitution machine for securing warfare communications [Kahn, 1967]. The machine known as the Enigma was used during the Second World War. The Enigma was an important tool for Germans and they trusted in its encryption power. This turned out to be a wrong assumption because as a sum of coincidences and hard work of Polish and English mathematicians and engineers "the

spell of the Enigma" was broken in the late 1930s. The sloppy handling of Enigma's keying and training material amongst Germans helped allied forces to advance with long leaps towards general solving method of messages encrypted with the 3 rotor military Enigma.

Figure 9. Germany's Enigma - Rotor based substitution cipher machine

Technically speaking the Enigma itself uses consecutive, changing monoalphabet substitutions to achieve polyalphabet substitution. The substitutions are accomplished with mechanisms known as rotors. Rotors are like changing mapping functions where an input letter is mapped to an output letter. The output letter of a rotor changes as the rotor rotates according to some machine-dependent rule.

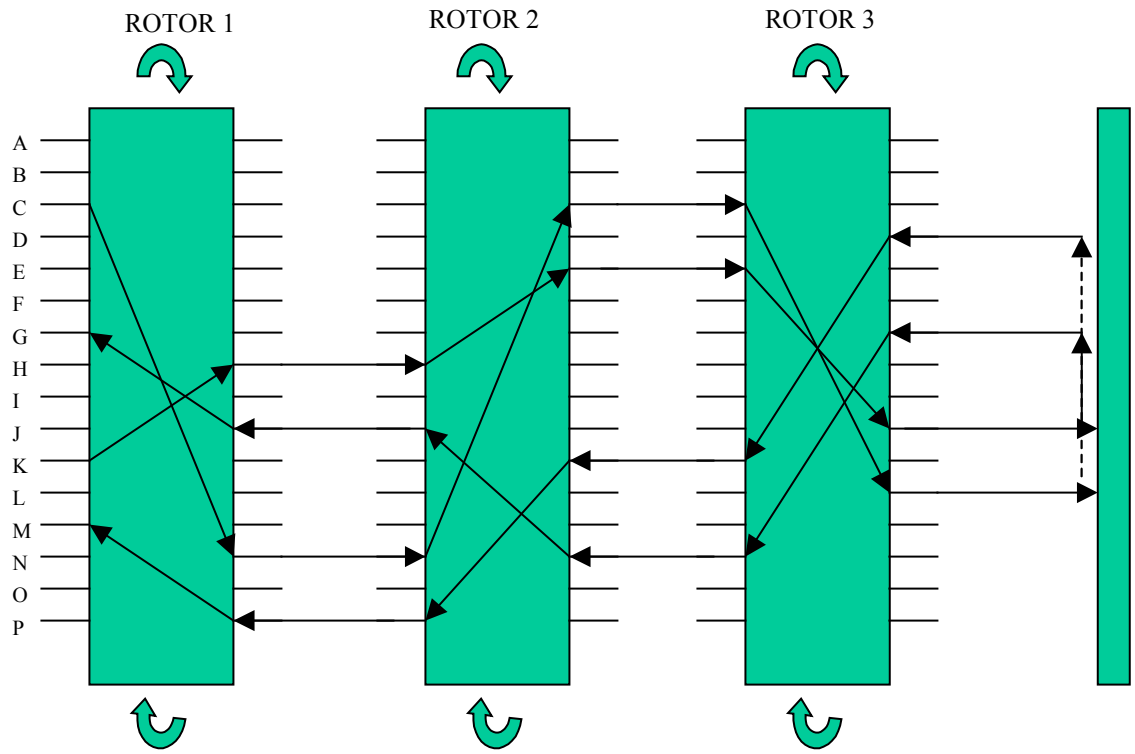


Figure 10. Enigma's example substitution diagram

We could say that the three rotor version of Enigma operates like a clock where the letters are like seconds. The first rotor rotates one step after each letter – like seconds. The second rotor rotates one after the first rotor has rotated a full round –like minutes. And the third rotor rotates one step after the full round of the second rotor – like hours.

The importance of secret keys is still valid today; The enciphering system can and should be public as long as it is mathematically complete, but the selection of the key, its storage and exchange between communicating parties has to be kept in secrecy.

3.2 Modern times

The concern for security services in digital information environments has evolved all the time. The need for digital information security has become more and more evident along the development of computer networks and the whole digital information infrastructure in general. The development race between new technologies and services using them is setting new requirements for information security continuously. Multicasting is one of the newer technologies that require special security arrangements.

Although the digital technologies evolve all the time the fundamental cryptographic goals have been the same throughout the time. Cryptography is only one way to provide information security but we restrict our discussion only to it, as it is sufficient for our needs regarding the multicast key management. In this section we take a brief look at

these security concepts in cryptography and show how they are suited to digital environments.

3.2.1 Security concepts

The main goal of cryptography is to build a framework with which the information can be secured. The building blocks of this framework are:

Confidentiality	Confidentiality provides mechanisms for keeping information from unauthorized entities.
Data integrity	Data integrity provides mechanisms for making sure that information remains unchanged.
Authentication	Authentication identifies an entity or affirms the origin of some information.
Non-repudiation	Non-repudiation prevents an entity from denying some performed transactions.

Cryptographic framework makes use of various mathematical tools to achieve the properties defined above. More formally these tools are known as security primitives.

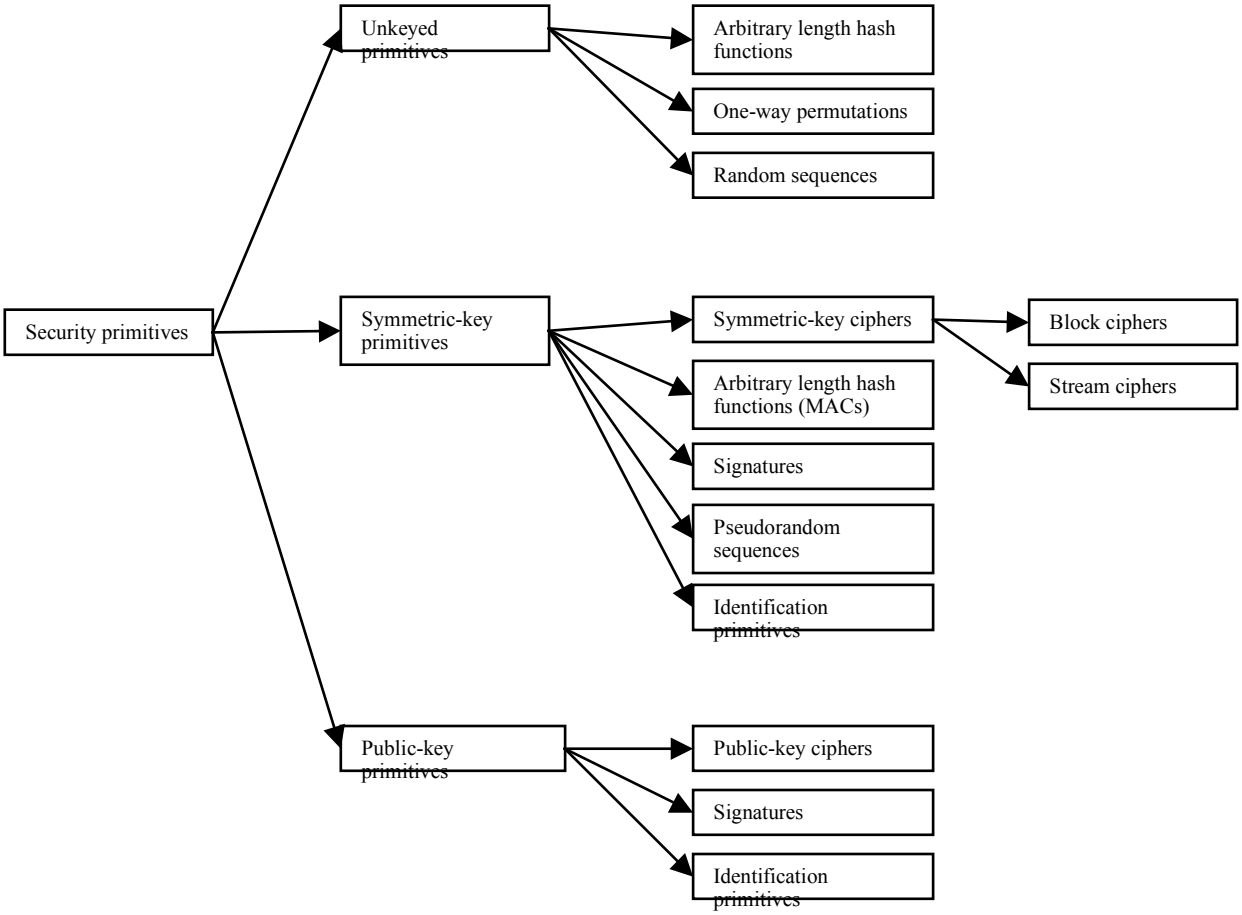


Figure 11 . Security primitives in cryptography

Some of these primitives perform overlapping functions from security point of view, but mostly their functionality is unique to each primitive. And together when their functionality is combined they form building blocks for information security framework. It is the matter of information security needs to define and select the correct security primitives to be used for each specific situation.

In this study we focus our attention mainly to symmetric-key and public-key primitives as they are mainly used to provide digital information security with cryptography.

Most strong cryptographical methods rely on well-known number theoretical problems. These methods are usually public and their effectiveness does not rely on obscurity. History has shown that we do not achieve good results with cryptographical methods that rely on the secrecy of the method itself. The only things that have to be kept secret are the secret keys used by these methods.

Before we can introduce security primitives we have to present a few fundamental cryptographical concepts.

Let A denote a finite set of alphabet of definition. The binary alphabet, where $A = \{0,1\}$, is a commonly used alphabet.

M denotes a set known as the message space. M consists of strings of symbols from the alphabet of definition. In cryptography an element of M is known as plaintext.

C denotes a set called the ciphertext space. C consists of strings of symbols from an alphabet of definition, which may differ from the alphabet of definition for M. An element of C is called a ciphertext.

K denotes a set called the key space. The elements of key space are known as keys.

Each element $e \in K$ uniquely determines a bijection from M to C, denoted by E_e. We call E_e encryption function. To make E_e reversible the function has to be a bijection so that we can reverse a unique plaintext from each distinct ciphertext.

For each $d \in K, D_d$ denotes a bijection from C to M. This is known as the decryption function.

An encryption scheme consists of a set $\{E_e : e \in K\}$ of encryption transforms and a corresponding set $\{D_d : d \in K\}$ of decryption transforms with the property that for each $e \in K$ there is a unique key $d \in K$ such that $D_d = E_e^{-1}$. This gives us $D_d(E_e(m)) = m$, for all $m \in M$. An encryption scheme is also known as cipher.

When one wants to define an encryption scheme she has to select the transformations for encryption and decryption and also the corresponding keys $(d, e \in K)$. Note that encryption and decryption can have the some key.

With these definitions we are able to develop security building blocks somewhat further. [Menezes, 1997]

3.2.2 Symmetric-key encryption

Symmetric key encryption is the most common encryption scheme for achieving bulk data encryption. As the name suggests, in most cases, symmetric-key encryption uses the same $(d = e)$ key for both encryption and decryption. To be more exact it can be stated that if you know *e* it is computationally feasible to determine *d*.

Symmetric key ciphers are generally divided into two categories: block and stream ciphers. Block ciphers are the most well-known symmetric key ciphers. Block ciphers break up the plaintext messages into equally sized string blocks and encrypt each block at a time. They suit well into environments where the transmission links are reliable. Block ciphers are once more divided into subcategories: substitution ciphers, transposition ciphers and their combinations, product ciphers. Block ciphers can be designed to achieve very high rates of data throughput, which is the major cause to their popularity. The efficiency is due to the fact that the symmetric key ciphers can use very short key lengths and still gain strong encryption.

Stream cipher encrypts traffic one bit at a time. Stream ciphers are useful in environments where transmission is error prone or buffering capabilities of the

processing units is very limited. For example in wireless environments stream ciphers are very practical.

The downsides of symmetric key ciphers include that the secret keys must remain secret at both ends. Also, the management of keys becomes laborious when the number of communicating parties becomes large, as the keys for every communicating pair have to be renewed frequently. [Menezes, 1997]

3.2.3 Public key encryption

Public key encryption relies on the usage of one-way functions. Unlike symmetric key ciphers public key encryption uses two different keys for achieving secrecy. The public key is used for encrypting the message and the other key known as the secret key is used for decrypting the message. This is very useful as we can give the public key to anyone who wishes to communicate confidentially with the owner of the corresponding private key. The public key combined with the public key cipher is like a one-way function and the private key provides the reverse function when used with the cipher.

Actually public key cryptography can also be used for other purposes than encryption. The private key can be used to generate unique hash values or signatures from any stringified message. The signatures can then be used to prove the authenticity and origin of the message in question.

Even though public key cryptography offers the same security services as symmetric key encryption it cannot be used alone as public key operations are computationally very expensive. But the forces of symmetric- and public key cryptography can be combined to provide very efficient security services.

3.2.4 Key exchange methods

Before secure communication can begin between entities they have to be mutually authenticated and they have to establish some shared secret, which can be used as cryptographic keying material. This shared secret can be used as a key encryption key (KEK) or content encryption key (CEK) for communicating parties.

The traditional solutions for establishing shared secrets are a physical meeting, phone call, letter or any other similar way of communication. For digital environments and especially static machines these methods are not of course suitable.

Diffie-Hellman exchange

In the 1970s Whitfield Diffie and Martin Hellman were focusing their research efforts on overcoming the need for physically transporting the shared secrets from one place to another. After several years of studying they released a paper in 1976 that introduced a

revolutionary method for negotiating shared secrets. The paper described the first solution for negotiating a shared secret between two communicating parties that have not met before by exchanging cleartext messages over an open communication media. The method known as Diffie-Hellman exchange used public key techniques for the first time[1] and its power relies on the well-known discrete logarithm problem. Finding logarithms in a finite field is very difficult: where $y = f(x) = g^x \bmod p$ is easy to calculate, its inverse $x = f^{-1}(y)$ is not computationally feasible to find today. The basic Diffie-Hellman does not authenticate the keys so there is a possibility for a man-in-the-middle attack. [Black, 2000]

The basic version of the exchange can be described as follows:

Liisa generates a large prime number p and a generator g and sends them to Pauli over an open channel. Both Liisa and Pauli choose random secrets

x, $1 \leq x \leq p - 2$

and

y, $1 \leq y \leq p - 2$,

respectively, which they do not reveal to anyone, not even to each other. Then they compute the public key values and send them to each other:

$Liisa - Pauli : g, p$

$Liisa \rightarrow Pauli : K = g^x \bmod p$ (Liisa's public key)

$Pauli \rightarrow Liisa : S = g^y \bmod p$ (Pauli's public key)

(Of course the first two messages could be combined so that the key is negotiated with only two messages, but it is separated for the sake of clarity.)

After this they compute the shared secret:

$SS = S^x \bmod p (= g^{yx} \bmod p)$ (Liisa)

$SS = K^y \bmod p (= g^{xy} \bmod p)$ (Pauli) [Menezes, 1997].

SS is the shared secret number not known by anyone else than Liisa and Pauli. SS can be used as a keying material for cryptographic algorithms for negotiating the actual content encryption key. As this exchange does not authenticate the origin of the exchanged messages in any way it is open for active attack. By only passively eavesdropping the wire the hacker cannot gain any useful information that could be used to compute the shared secret.

There are many enhanced versions of the Diffie-Hellman key exchange available that authenticate both the keys and the origins of the messages or actually use a separate authentication before Diffie-Hellman exchange to authenticate the exchange.

---

[1] In fact Ralph Merkle also discovered a public key cryptosystem at the same time, but the printing of his paper was delayed for some reason and it was not released until 1978. Considering this fact, these three men Diffie, Hellman and Merkle are considered to be the fathers of public key cryptography today.

Digital signatures

Authenticated exchanges rely typically on the usage of digital signatures. Digital signatures take once again advantage of public key cryptography. Secret keys combined with one-way functions can be used to calculate unique hash values on any data. Hence the counterpart public key can be publicly available in some form so that the receiver of the message can confirm the authenticity of the data and the origin of the message.

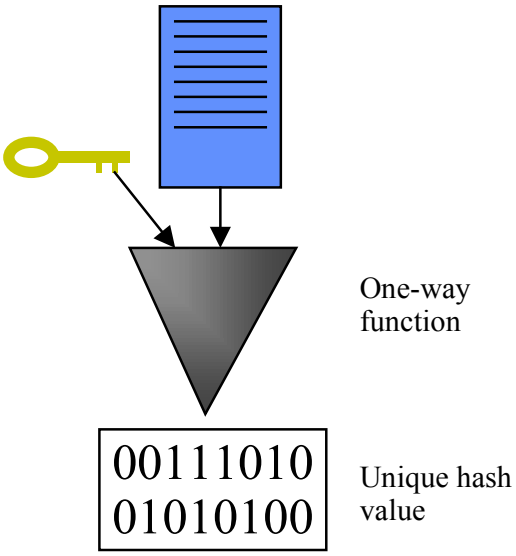


Figure 12. Digital signature

4. Multicast key management

In near future the networks will have bandwidths ranging from kilobits per second to gigabits per second. Networks will have different users and applications in heterogeneous network environments. All these entities will have different quality of service (QoS) and security requirements.

Secure multicast communication requires that the messages between group members are authenticated, encrypted and integrity checked. The group members want to be sure that their communication is not eavesdropped and that they are sure whom they are really communicating with. The requested quality service level has to be guaranteed even though security services are in use in the network. And when the composition of the group is changing the group members and service providers want to be absolutely sure that members leaving the group cannot read the messages.

4.1 Multicast group characteristics

Canetti and Pinkas [Taxonomy, 1999] have defined properties that can be used to quantify multicast groups based on their characteristics.

1. **Group size:** Small discussion groups might have some dozens of participants whereas virtual conferences and classes might have several thousands of participants. And massive concert broadcasts might have several millions participants.
2. **Member characteristics:** Member's computing power sets requirements for the amount and type data to be sent. It is also important to know whether the members are always present or is their presence occasional.
3. **Membership dynamics:** The group membership may be known in advance, or the members can join whenever they want and they can leave whenever they want. Members may be joining and leaving in bursts.
4. **Expected lifetime:** The group lifetime may vary depending on the type of the group; lifetime may vary from minutes to days and even unbounded amounts of time.
5. **Amount and types of senders:** Typically in broadcast type multicasts have only one sender. Then there might be several senders. Or some senders might generate significantly more traffic than others.
6. **Volume and type of traffic:** The communication may be heavy and it may need to arrive in real time. There may be maximum allowed latency for traffic.

All these characteristics have impact on the selection of the key management scheme, used cryptographic algorithms and needed authorisation services. Security issues should be considered right from the beginning of system design. Service providers should consider what are the most valuable assets and services they wants to protect within their business and evaluate how much they should invest considering the value of the business.

4.2 Multicast security issues

Canetti and Pinkas [Taxonomy, 1999] have also gathered security and trust related issues that should be considered when a multicast service is designed. All the concerns are not equally important to all types of services but their presence should be considered when a service is designed.

Group management and access control: It should be possible to make sure that only registered users are able to access the communication destined to the group. In some types of services it is needed that only the members of the group are able to send data to the group. Potential group members should be authenticated. It should be possible to distribute security keys only to current group members. It should be possible to revoke the membership of a leaving member. In some occasions the past communication of the group should not be accessible to new members. The security keys should be periodically refreshed to maintain better security.

Ephemeral secrecy: Sometimes a mechanism that only delays access for some amount of time may be sufficient. Or sometimes it is enough to prevent access to some certain part of data. For example with digital media streams it may be enough to encipher only some part of the stream to make the stream unviewable to non-group members.

Long-term secrecy: Some services may handle information that should remain confidential for really long time or even eternally.

Perfect forward secrecy: Even if some security keys are compromised at a later date it should not be possible access the data encrypted with newer keys.

Sender and data authenticity: In multicast environments the authentication can happen in two levels. First, the authentication may only need to make sure that the data is received from some group member. Or secondly there may be a need to authenticate the messages from each member individually.

Anonymity: In some services it may be desirable for users to stay unknown. Some services may support the anonymity of the sender only. Then there may be applications that need to protect the anonymity of every user of the group – careless of the role of the group member.

4.2.1 Multicast piracy

The purpose of multicast security is to provide security services inside the multicast group. On the other hand nothing stops some authorised users from co-operating with some unauthorised third party to receive group messaging in cleartext. This kind of arrangement is a common scene in the normal television-broadcasting environment. A lot of effort has been put to fight this kind activity in the cable-TV industry, but this problem is really very hard to overcome.

In multicast environment this kind of piracy would mainly concern one-to-many multicast services. The only way to oppose this kind of activity is to develop effective means for spotting piracy activities and remove the rotten members from the group.

4.3 Multicast security issues - Example

Our example company Keytouch communications provides multicasting services where each service requires some kind of security. We now inspect Keytouch's services more in detail and determine which of the Canetti's and Pinka's multicast security issues apply to each service.

PayTV is a typical one-to-many multicasting service. It sends multimedia streams to the registered receivers, which form the multicast group itself. The members have been registered and authenticated by Keytouch.com. Ephemeral secrecy is applied to

multimedia streams, as we do not want any other than eligible users to view the broadcasts.

Secure conferencing applications are a bit more complicated compared to PayTV. The nature of conferencing application assumes that the traffic is many-to-many and that the traffic is encrypted and authenticated. Official corporate type conferences require that the identity of each group member be used as the basis for authentication whereas entertainment type chat conferences may require only group level authentication or no authentication at all. Anonymity is also a common requirement for entertainment chat conferences.

The user of an information delivery service should always be sure that (s)he is receiving authentic information. Also, the information should be encrypted to restrict unauthorized access.

The polling service is conceptually and technically the most complex service provided by Keytouch.com. Polling service must be able to identify a user and register when (s)he has used his/her vote, but the system should not provide any hints to anyone about what the user voted for. Yet the system should still be able to register the user's opinion and provide information to the polling group members about the progress of the voting.

4.4 Group key management

As multicasting optimises the usage of network links by sending one packet to the whole multicast group instead of sending a separate packet to each recipient, it is very natural to encrypt the multicast packets only once too. This leads us to the introduction of the concept of the group key. A group key is a common security key that is distributed to the whole secure group. Group key can be used to encrypt a multicast packet and the valid multicast users are able to decrypt the packet using the group key. The usage of group key itself is very simple, but the distribution and revocation situations are problematic.

When a member of a multicast group leaves the group all the common group keys have to be renewed so that the leaving member cannot receive any later messages destined to the group. As we do not have any more secure group keys which we could use to distribute the new group keys to each member we have to use member specific keys or some other way to create a new group key. From performance standpoint the key renewal operation becomes very heavy if the number of group members is very large and only one server is responsible for doing it. To distribute new keys to every member separately is very ineffective.

In general we are able to identify characteristics for key distribution system that should be considered when we are selecting or defining a key management scheme [Moyer et al., 1999, Taxonomy, 1999].

Number of keys stored by the controller: In a simple, non-scalable scheme the controller has to store *n+1* keys for a group size *n*: the group key and personal security key or Key Encryption Key (KEK) for each member. In more powerful schemes the controller has to store more keys.

Number of keys stored by a group member: With the simple solution group members have to store only two keys each, but the bandwidth requirements for this solution are intolerable. In more efficient schemes the members have to store more keys to gain bandwidth efficiency.

Join and Leave secrecy: Join secrecy assures that new members cannot read the past messages of the group. On a join event the group key is renewed to the whole group to ensure this. Join secrecy is typically easy to achieve as we can make use of the old group key to distribute the new group key to the old members and to the new member using the personal key encryption key. Leave secrecy guarantees that the leaving members cannot read the future messages of the group. This is usually a more resource hungry operation, as we do not have any group secret that we could use to securely multicast new group secret to the whole group. But we still have to have some key or other mechanism that enables controller to communicate with the valid members securely. Poovendran et al. [Poovendran and Baras, 1999] call this property *reachability condition*.

Number of messages needed to update keys on a join: Join operation is not typically as greedy for the number of messages as the leave operation is. We may even do some sacrifices here for the sake of a more optimised leave operation.

Number of messages needed to update keys on a leave: Leave operation with the simple solution requires *n* messages to be sent by the controller. We are able to optimise this at the cost of key storage space by subgrouping users as we will see later.

5. Comparing existing key management schemes

We now present some key distribution scheme alternatives and compare their properties. We show first a simple centralised straightforward solution. Then we present schemes where some of the management tasks are delegated to trusted third parties. Then we show how hierarchical key tree can be used to make key distribution more efficient. And finally we end our tour with a totally distributed management solution where each member gets to contribute to the construction of the group key.

5.1 Group key management protocol (GKMP)

The most straightforward way to distribute keys is to use a centralized group key controller (GC), which takes care of the group key management. Every time a member leaves the group a new group key has to be generated and distributed to the group members. The new group key has to be encrypted with member's personal key encryption key (KEK). The distribution cost is linear to the group size. In addition the controller has to store *n+1* keys. Also, if the centralized controller fails it is fatal to the whole system.

The simple key management seems tempting in its simplicity, but it cannot really be put into any serious use. Every time a member joins or leaves we have to distribute new group keys to every single member, as we want to protect the integrity and secrecy of past and future messages destined to that group. For large, constantly changing groups we definitely need sub-linear cost distribution mechanism. Large groups may produce heavy load for the key controller, which affects its availability and ability to serve. The fault tolerance that the simple management system cannot provide due to its centralized nature is also vital for large groups. [GKMPA, 1997] [GKMPS, 1997]

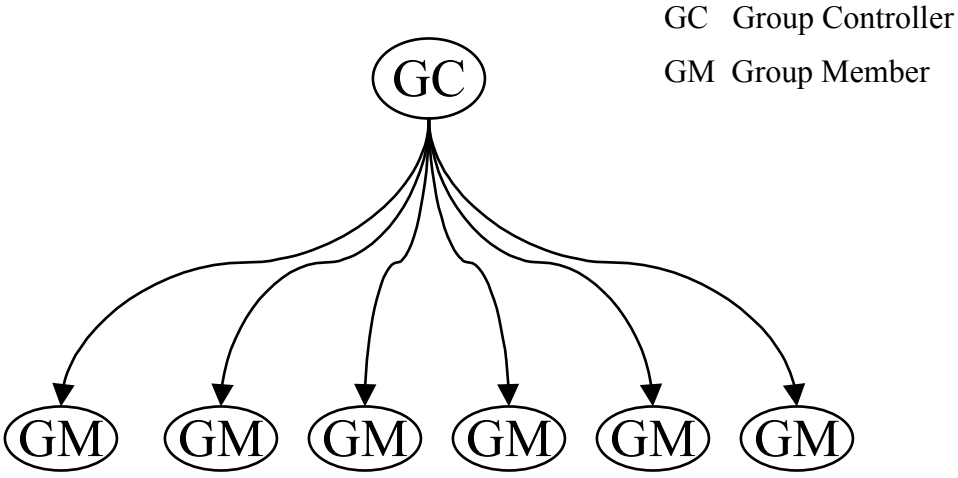


Figure 13. Group Key Management Protocol

5.2 Scalable multicast key distribution (SMKD)

Key distribution mechanism becomes a bit more scalable and more fault tolerant if we distribute authentication and encryption tasks. SMKD makes use of the multicast distribution tree. SMKD is based on Core-Based Tree (CBT) routing protocol and it is able to delegate encryption and authentication tasks to downstream routers. This means that each router knows its directly attached downstream neighbours and it is authorized to negotiate keys with new hosts and routers under it. [RFC1949, 1996]

Initially when the group is started the core router is the group controller. As routers join the multicast delivery tree they are authorized to act as a group controller for their directly connected subtree. [RFC1949, 1996]

This approach is better than the GKMP from scalability perspective, but this is still impractical when the distribution group becomes large. SMKD is also dependent on the underlying multicast routing protocol.

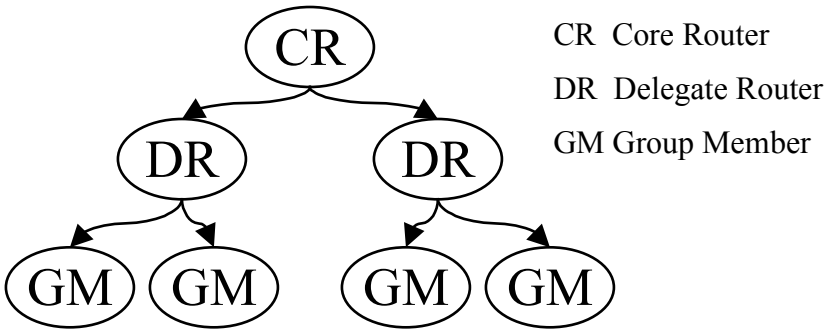


Figure 14. Scalable multicast key distribution

5.3 Iolus

The Iolus key management scheme takes a similar scalability approach compared to SMKD. Iolus uses secure distribution tree to link users to a single group. In practice the multicast group is divided into subgroups. At the top level there is a Group Security Controller (GSC), which controls the subgroup controllers known as Group Security Intermediaries (GSIs). Both GSCs and GSIs can also be called with a common name Group Security Agents (GSAs). [Mittra, 1997]

Each subgroup has its own security keys. Each group's GSI manages the key for the subgroup. GSI acts as a mediator between its subgroup and other subgroups. GSI generates key for the subgroup and takes care of the registrations to the upper level in hierarchy. GSI also delivers messages from and to the other subgroups. [Mittra, 1997]

We take a closer look at some of the main operations involved in Iolus framework. We inspect how the group can be initialised, how members can be added and deleted and how the data transmission between subgroups is managed.

Initialising the group and adding members to it

When there is no members in the group there is no connection between GSIs and GSC. When a strongly authenticated user wants to join the group it contacts a GSI and sends a join request. The GSI generates a group key and sends it using a secure unicast message. After that it sends a join message to the upper level GSI. This goes on as long as the upper level GSA is GSC. When GSC is finally contacted the user becomes the member of the group. If the GSI contacted by the user or some upper level GSI is already a member of the group there is no need to contact upper levels and the user becomes a member of the group.

When a GSI contacted by the new user has already members in its subgroup the key renewal for the whole group requires only two messages and encryptions. The new group key for the new member is encrypted using the member's private key. The new group key for the rest of the subgroup is encrypted using the old group key and distributed using multicast. This kind of approach does not provide perfect forward secrecy for the future communication, but is secure enough for most purposes. [Moyer et al., 1999]

Deletion of a member

A member can request deletion from the group or (s)he can be thrown out for some reason. If there are no other members in the subgroup the GSI contacts its parent and removes itself from the distribution tree. When there are several members in the subgroup and a member wants to leave the group key has to be renewed. As we cannot use the old group key for renewal this requires again a linear amount of messages and encryptions [Moyer et al., 1999].

Data transmission

Every subgroup has its own group key and no other subgroup knows it. If a member within a subgroup sends a message to the group (s)he encrypts it with the group key and every member including GSI in its subgroup knows how to decrypt it. GSI's job is then to deliver the message to other subgroups by first decrypting the message with its own subgroup's group key and then encrypt it with the upper level group's key.

For example in Figure 15 we have a part of the Keytouch.com's secure conferencing application's distribution tree. C2 wants to send a message to the whole group. It encrypts the message with subgroup A's key. C1,C3 and GSI_A are able to decrypt the message. GSI_A then encrypts the message with subgroup B's key and sends it to the subgroup B. GSI_B does the same tricks and forwards the message to the subgroup K. This way every GSI decrypts the message and encrypts it with the upper level's group key and GSIs receiving messages from upper level decrypt the message and encrypt it with its own subgroup's key. In this fashion the message gets transmitted to the whole secure conferencing group. [Moyer et al., 1999] [Mittra, 1997]

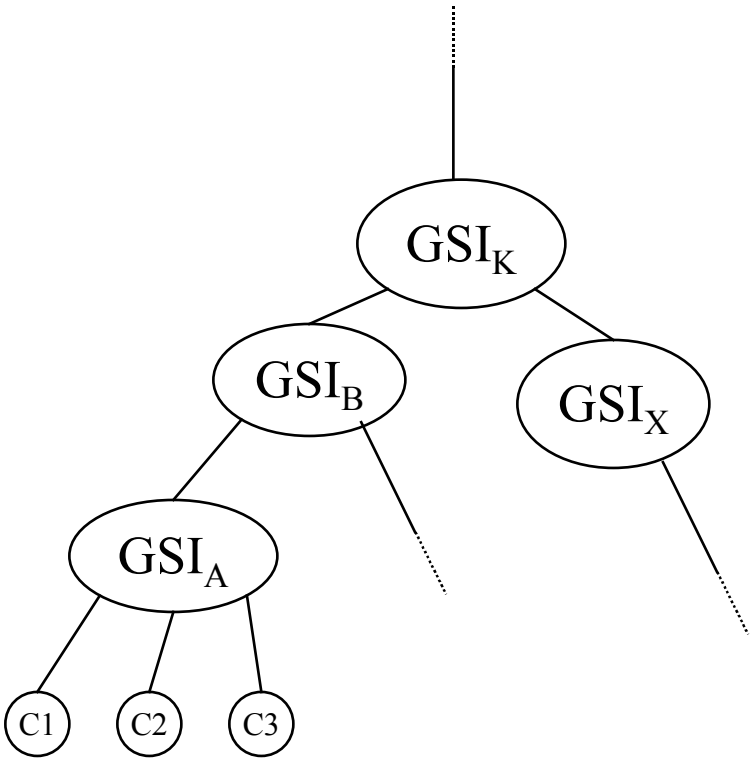


Figure 15. Iolus data transmission example

Iolus framework distributes wisely the workload caused by the group topology changes to the GSIs. Even if the subgroup does not use any special distribution mechanism for distributing keys the operations are still powerful as the groups are typically small. The delegation of group management tasks also helps to avoid the problems associated with centralised group management. Of course the GSC is still a single point of failure, but when a GSI suffers from a system failure only the subgroup in question is left without service.

The downsides of Iolus include that the group operations are very resource intensive and the amount of group management overhead increases as the number of GSIs grow when the group becomes larger. In addition the transmission of messages requires decryption and encryption operations at each GSA. And as mentioned earlier the centralized nature of GSC jeopardizes the operation of the whole group in the case of failure.

5.4 Complementary key approach

Complementary key scheme (CKS) optimises the key management bandwidth usage at the cost of key storage space. CKS takes also a tree based approach where it has a root controller which shares a separate key encryption key (KEK) with each member or leaf i where $i \in [1, N]$. Root generates the group key for the multicast communication and distributes it separately to each leaf *i* encrypted with KEK(i).

This scheme is called complementary because the root generates something known as complementary variable for each leaf and sends them to the leafs. The root will not send leaf's own variable, but it sends the variables of all the other members – the complementary variables. Hence every leaf has to store N+1 variables: the KEK, N-1 complementary variables and the group key.

These complementary variables become useful when a leaf is leaving the distribution tree. As the leaving leaf does not possess the complementary variables of its own the others can form a new group key only by knowing the index of the leaving member and some common deterministic way to generate the secret. Basically the root has to only send one cleartext multicast message indicating the index of the leaving member compared to sending a separately encrypted unicast message containing the new group key to each leaf.

From transmission bandwidth perspective with this scheme we can gain large savings, but as the group becomes large the (N-1) stored complementary variables can begin to take up huge amounts of space in each leaf. Then again if several members are being deleted from the group at the same time they may have collusion intensions. They can exchange complementary variables to gain access to the group traffic again.

[RFC2627, 1999]

5.5 Hierarchical tree approach

Two separate research groups independently discovered hierarchical tree schemes. Both Wallner et al. [RFC2627, 1999] and Wong et al. [Wong et al., 2000] noticed that when we assign several keys per user we can find a well performing key management scheme and balance between storage space, number of message transmissions and key encryptions.

The keys are organised in a k-ary tree so that the internal nodes of the tree hold a key and hence some keys are common to several users in a similar manner as in Iolus [Mittra, 1997]. Thus, each member knows a subset of all key including a personal key encryption key (KEK). The keys are organised so that each user knows the keys from leaf - itself to the root, but no other keys.

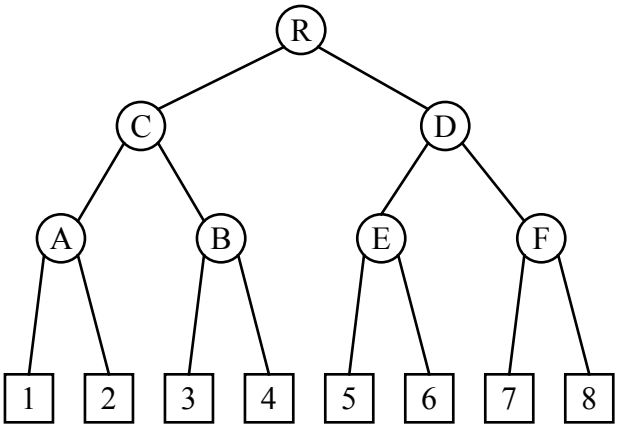


Figure 16. Hierarchical key tree

Wong et al. [Wong et al., 2000] have taken a formalised approach to present secure groups. A secure group can be defined with the triple (U, K, R) where:

- U is a finite set of users,
- K is a finite set of keys,
- R is a binary relation between U and K, $R \subset U \times K$. This is known as the user-key relation of the secure group. User u has key k if and only if (u,k) is in R.

Messaging between principals can be described with the following notation:
A message m encrypted with key K from a to b can be written
$a - b : \{m\}_K$.

Every secure group has a key controller, which generates and distributes the keys. Key controller is responsible for maintaining the relation R between U and K as the group topology changes.

The relation R can be presented using key graphs [Wong et al., 2000]. A key graph is a directed acyclic graph G where the vertices reflect the elements of $u \in U$ and $k \in K$. The edges represent the relation R. U-vertices have only outgoing edges whereas k-vertices may have both incoming and outgoing edges. When a k-vertex does not have any outgoing edges it is called the root.

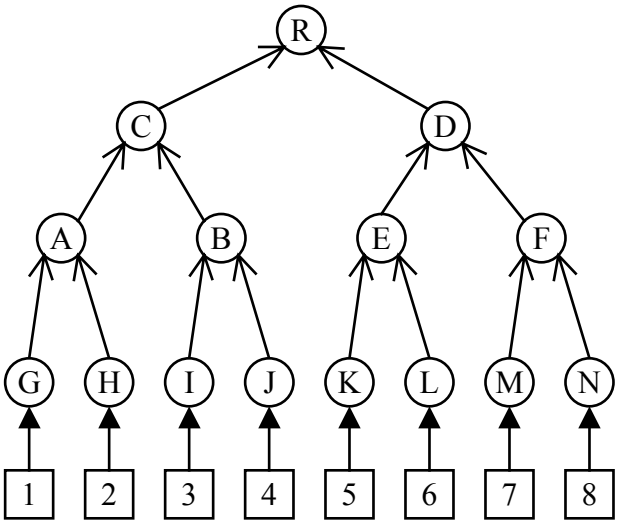


Figure 17. Key graph

The key graph in Figure 17 is based on the following sets:

$U = \{1,2,3,4,5,6,7,8\}$

$K = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, R\}$

$R = \{(1, G), (1, A), (1, C), (1, R), (2, H), (2, A), (2, C), (2, R), (3, I), (3, B), (3, C), (3, R), (4, J), (4, B), (4, C), (4, R), (5, K), (5, E), (5, D), (5, R), (6, L), (6, E), (6, D), (6, R), (7, M), (7, F), (7, D), (7, R), (7, N), (7, F), (7, D), (7, R)$

Let us define two helpful functions for finding subsets in relation R:

$keyset(u) = \{k \mid (u, k) \in R\}$

$userset(k) = \{u \mid (u, k) \in R\}.$

[Wong et al., 2000]

The function *keyset* finds the set of keys, which belong to user u in U. And the function *userset* finds the users that know the key k in K.

Special categories of key graphs

The shape and size of key graph deserves special attention and next we will inspect some specially structured key graphs to see how their attributes make a difference in key management.

Star graph

Star key graph is a special type of secure group where each user in U has only two keys. These are the personal KEK and the common group key. We will now illustrate the join and leave operations using star form key graph.

Join operation

Consider the situation shown in Figure 18. When user *8* wants to join the group (s)he sends a join request to the key controller *c*. The controller authenticates the user and grants a permission to join the group. When we are adding users to a star graph the new user is attached directly to the root key vertex so there is no need look up a k-vertex where to add the user. User *8* negotiates a personal KEK with *c* and *c* generates a new group key and distributes it to the whole group. The key graph before and after join can be seen in Figure 18 a) and b) respectively.

More formally *c* has to send the following messages to renew group key:

(1)$c - \ 8 : \{R_2\}_N$

(2)$c - \ 1...7 : \{R_2\}_R$

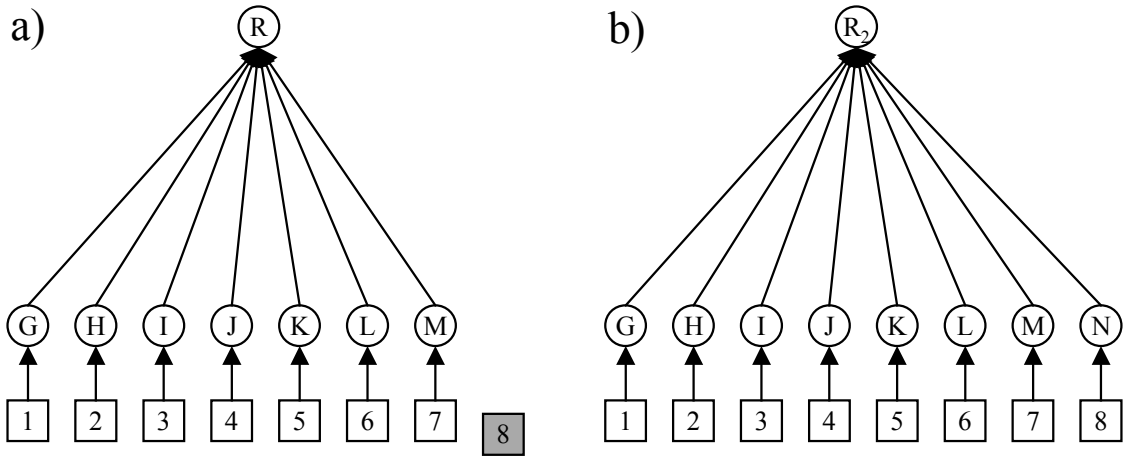


Figure 18. Star key graph join

Leave operation

Assume that in the situation of Figure 18 b), user *8* sends a leave request to the controller *C*. *C* removes the KEK(N) and generates a new group key R. *C* has to distribute new R using each user's individual KEK (G,H,I,J,K,L,M).

(1) c generates a new group key R

(2) for each member in U-{8} do

$$c - \ i : \{R\}_{KEK(i)}.$$

Star graph's bandwidth and storage requirements are identical compared to GKMP presented earlier. Thus, star key graph is only suitable for small groups as its scalability for larger groups is really bad due to O(n) message cost in leave operation.

Tree graph

Tree key graph is a special type of secure group in that it has only one root. The tree graph can be parameterised with two parameters:

- The height *h* of the tree is the number of edges in the longest directed path in the tree.
- The degree *d* of the tree is the maximum number of edges of a vertex in the tree.

The values of these parameters are crucial for getting the best performance in use from hierarchical key trees.

We will now show how join and leave operations are performed using tree key graphs.

Join operation

Assume that in the situation of Figure 19 a), user *8* makes a join request to the controller *c*. Controller *c* authenticates *8* and grants a permission to join the group. Then *c* finds a suitable joining point k-vertex where *8* can be attached. Now *c* has to generate new keys from the joining point to the root to inhibit *8* from accessing past communication. Then *c* distributes securely the necessary keys to members as needed.

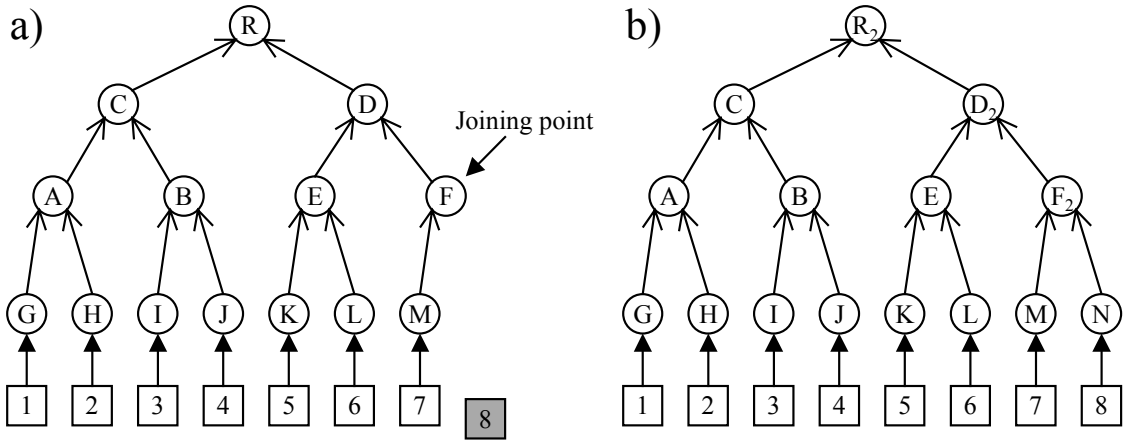


Figure 19. Tree key graph join

If we look at the Figure 19 we notice that the keys in vertexes F, D and R populating the path from joining point to the root have to be changed. *C* has to find the minimum set of users for each key in order to optimise the update messaging. Wong et al. [Wong et al., 2000] have proposed three different key distribution strategies as how the controller should construct the rekeying messages: *user-oriented rekeying*, *key-oriented rekeying* and *group-oriented rekeying*. With these strategies we are able to achieve different kind of optimisations regarding the bandwidth usage, controller and user processing load although one of these strategies has a superior overall performance.

User-oriented rekeying

In the user-oriented rekeying the controller constructs each rekeying message so that they contain exactly all the messages that some user or a group of users need. For our example the user-oriented messages are constructed as follows:

(1) $c - \{1,2,3,4\} : \{R_2\}_R$

(2) $c - \{5,6\} : \{R_2, D_2\}_D$

(3) $c - \{7\} : \{R_2, D_2, F_2\}_F$

(4) $c - \{8\} : \{R_2, D_2, F_2\}_N$.

Key-oriented rekeying

Key-oriented strategy emphasizes that each new key should be packed into a separate message and distributed to the holders of the old key. For example key X is to be replaced with a new key X'. Hence the recipients of the message can be apprehended with *userset(X)*. This approach excludes of course the new user of the group. The key-oriented messages are as follows:

(1) $c - \{1,2,3,4,5,6,7\} : \{R_2\}_R$

(2) $c - \{8\} : \{R_2\}_N$

(3) $c - \{5,6,7\} : \{D_2\}_D$

(4) $c - \{8\} : \{D_2\}_N$

(5) $c - \{7\} : \{F_2\}_F$

(6) $c - \{8\} : \{F_2\}_N$.

Group-oriented rekeying

In group-oriented rekeying strategy each new key is encrypted separately into large messages or one large message destined to subgroups or the whole group. This means that many users will receive large messages including keys that they do not need. This kind of procedure still has some advantages over user-oriented and key-oriented strategies. If we are distributing one large message to the whole group we can use multicasting. Also, this causes a lot less messaging overhead for the controller's rekeying messages. And the total number of transmitted bytes by the controller is a lot less with this kind of approach. The group-oriented messages can be constructed as follows:

(1) $c - \{1,...,7\} : \{R_2\}_R, \{D_2\}_D, \{F_2\}_F$

(2) $c - \{8\} : \{R_2, D_2, F_2\}_N$.

Leave operation

If the user *8* shown in the Figure 20 a) is to be deleted from the tree key graph for some reason or another the controller *c* deletes first the user *8*'s u-vertex and the k-vertex for the personal KEK. Then *c* generates and distributes the new keys as needed. All the

keys that were known by the user *8* from the joining point to the root have to be renewed. In the example Figure 20 b we have to renew the keys F_2, D_2 and R_2.

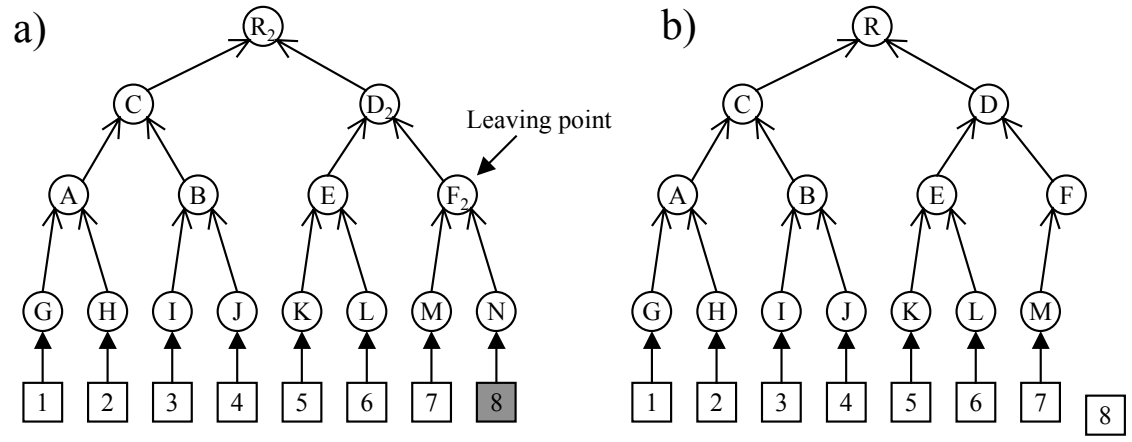


Figure 20.Leave from key tree graph

We will now describe the rekeying operations using the *user-oriented*, *key-oriented* and *group-oriented rekeying* strategies.

User-oriented rekeying

Each user receives exactly the keys they need in a single message encrypted with some key with which they can be reached. The server optimises the operation by finding the minimal number of keys that are needed to encrypt the packets securely.

The messages are as follows:

(1) $c - \{1,2,3,4\} : \{R\}_C$

(2) $c - \{5,6\} : \{D, R\}_E$

(3) $c - \{7\} : \{D, F, R\}_M$.

Key-oriented rekeying

The principle here is also the same as in join operation: all the keys from the *leaving point* F_2 to the root have to be renewed. Each new key k' is sent in a separate message to the holders given by userset(k) encrypted with the old key k. Let's consider our example in Figure 20 a) and b) again. The following messages are sent:

(1) $c - \{1,2,3,4\} : \{R\}_C$

(2) $c - \{5,6\} : \{R\}_D, \{D\}_E$

(3) $c - \{7\} : \{R\}_F, \{D\}_F, \{F\}_M$.

Group-oriented rekeying

In group-oriented leave operation we construct a single rekeying message containing all the keys to be renewed. We try to use subgroup keys as much as possible and we

multicast the single message to the whole group. The leave event in Figure 20 b) and a) the message is as follows:

(1) $c - \{1,2,3,4,5,6,7\} : \{R\}_C, \{R\}_E, \{R\}_M, \{D\}_E, \{D\}_M, \{F\}_M$.

We have now illustrated all the three rekeying strategies for both join and leave operations. The common goal for all the approaches is to minimise the amount and size of messages sent be controller to members In our examples we have had trees with degree two, but it can be shown that the optimal degree of a key tree graph is four [Wong et al.].

Hierarchical trees seem to be the most attractive solution for large-scale multicast applications.

5.6 A Distributed Framework for Scalable Secure Many-to-Many Communication (DISEC)

DISEC is a totally distributed multicast key management scheme that does not make use of any centralized controller. It is best suited for many-to-many communication where most of the group members are multicast sources. The main idea in DISEC is to distribute the key management tasks and overhead evenly between group members as opposed to many other mechanisms where the controller does most of the work.

DISEC uses a virtual binary tree for managing key distribution. The tree consists of virtual internal nodes and the actual member nodes as leafs. Each member generates a secret key, which is known as the unblinded key. The blinded key is created from the unblinded key using a one-way function, which means that it is computationally unfeasible to try to calculate the unblinded key from the blinded one. The keys for the internal nodes are then generated using the blinded versions of the members' secret keys such that a parent node's secret key is calculated from the two blinded keys of its children with help of a mixing function. Thus all the member nodes contribute with their key to the forming of the root key. This way each member has to generate and know all the blinded and unblinded key on the path to the root and also all the blinded keys of the siblings of the nodes on the path to the root. The idea of virtual binary trees and one-way function trees where each member get to contribute to the forming of root key was first introduced by McGrew et al in [McGrew, 1998] and has been adopted to DISEC from there.

Each node is assigned a unique binary identifier (ID) that is used to form a *key association*. The identifiers are constructed using a common variable-length binary

coding method. The root does not have an identifier, but from there on every left child of a node adds a '0' to the parent's identifier whereas a '1' is added to form the right child's identifier. When a new member joins the group we can select identifier with based on local information without the need for any centralized controller. A virtual binary tree can be seen in the Figure 21.

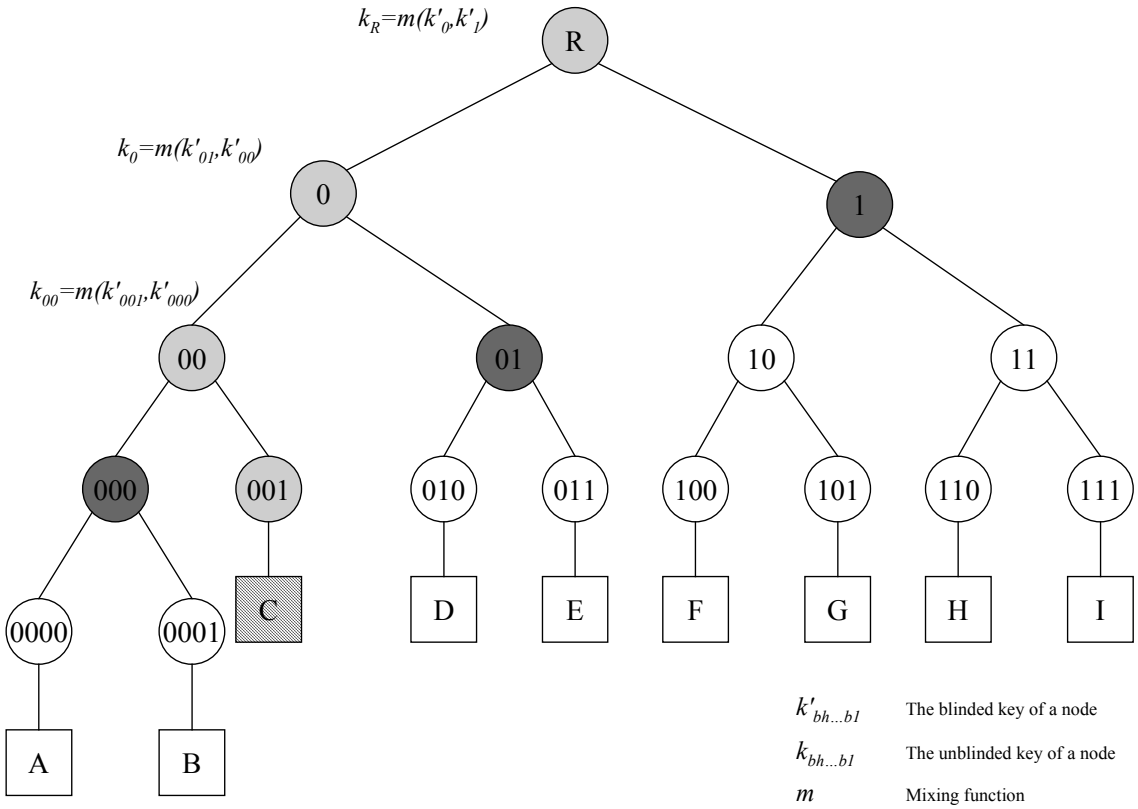


Figure 21. A DISEC virtual binary tree with binary identifiers

The structure of the identifier has some significant properties that are used when we are constructing the distribution key hierarchy. The identifiers can be used to track down efficiently the member's key association group and neighbours, as they have to be known before we can construct all the needed keys. The key association group and neighbours can be discovered using simple bit manipulation operation based algorithms. The detailed algorithm description is out of the scope of this study and can be found from Doneti's arcticle[Doneti, 2000].

A member has to generate a key – the root key - that is common for the whole group before it can send any secured multicast messages. Before the root key can be constructed the member has to generate and exchange certain blinded keys with other members of its key association group. Key association group consists of the members that can provide the other blinded keys used with the mixing function to generate the unblinded keys for a node.

For example in Figure 21 node *C* has to query blinded keys from key association group members *A*, *E* and *G* and generate the unblinded keys for nodes *00* and *0* before

it can generate the unblinded root key for node R. The blinded keys for node *00*'s mixing function are the keys of nodes *000* and *001*. One blinded key for node *00* is generated by the node *C* itself and the other is queried from node *A*. Node *C* sends also the blinded key *001* to *A*. Node *B* knows also the needed blinded key, but the key association group algorithm returns always a single member for each needed key. The same procedure happens with node *0*; *C* generates the blinded key for *00* and queries the other needed blinded key *01* from member *E*. Then the unblinded key for *0* is created. After that *C* queries the blinded key for *1* from member *G* and generates the root key *R*.

The distributed nature of this scheme causes that each member has to exchange the blinded keys with their own key association group members and generate and store the unblinded and blinded keys as described above. We may of course make use of subgroup multicasting to distribute the blinded keys more efficiently so that we do not need to make as many unicast exchanges between members. The rekeying procedure is initiated every time the group topology changes. We will now give short descriptions on how the actual topology changes are handled in DISEC.

Join operation

When a user desires to become a member of a DISEC multicasting group (s)he contacts some member of the group. The member checks the user's credentials and makes the needed changes to the virtual tree and initiates rekeying. It is advantageous to try to keep the tree as balanced as possible when topology changes occur. Without any centralized controller and without any global topology information this is a bit tricky. However in join operation the prospective user may choose to contact the member with the shortest ID length to keep the tree balanced.

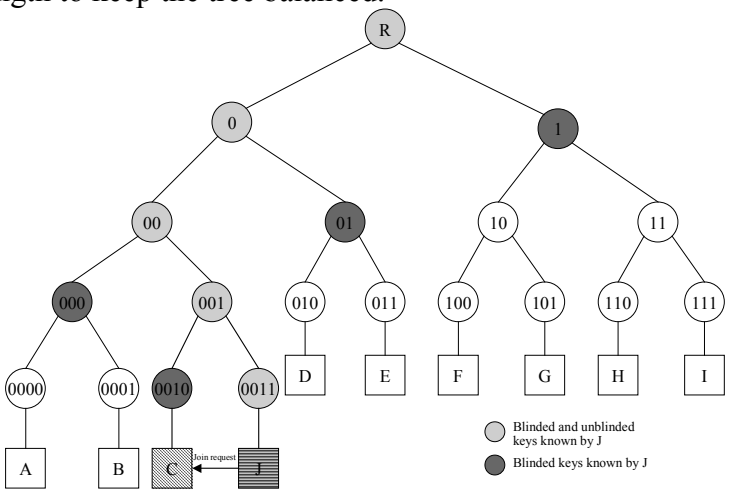


Figure 22. DISEC join operation

In Figure 22 *J* has sent a join request to *C*. After authenticating *J, C* divides its binary ID into two new ones by following the previously described guidelines: a '0' is added to the existing parent ID for the left child *C* and a '1' for new right child *J*. Then

both *C* and *J* generate new secret keys and exchange the blinded versions of them with each other. The change of secret keys at the member level means that the whole key hierarchy has to be constructed again. Altogether *O(log n)* unicast messages and *O(log n)* multicast messages are sent during join.

Leave operation

When a member leaves the group the key hierarchy has to be partly rebuilt again - new keys have to be agreed with the key association group and certain binary IDs have to be assigned differently. The rekeying procedure is the same as in join, but the ID assignment deserves some attention here. If the neighbour of the leaving member is a sibling it assumes the binary ID of its parent. Another case is when a neighbour whose sibling is leaving has descendants. For example in Figure 21 the member *C* is leaving the group. The sibling's descendants *A* and *B* are notified and they shorten their binary ID by one bit so that IDs become 000 for *A* and 001 for *B*. Altogether *O(log n)* unicast messages and *O(log n)* multicast messages are sent during leave operation.

Summary

A definitive advantage of DISEC is its distributed approach for key and group management and ID assignment. DISEC is also very scalable and it is suitable for example for very large discussion group applications and other similar many-to-many multicast applications. Except for the join and leave notifications there isn't any kind of control messaging between members, which supports the distributed nature of the scheme. This scheme is also resistant against collusion between members as the keys are based on contributory material. Also, network or controller failures do not affect this scheme as the group can continue communication even if some parts of network are not functional.

5.7 Summary of the presented schemes

All the key management schemes presented in this chapter achieve the same result with different approach. The key motivators behind these approaches are some specific type of multicasting or even some specific application area. A gap between generations can be clearly seen in the presented schemes: The hierarchical tree based schemes and truly distributed one-way function tree based schemes represent the newer breed. The foundation work has been of course done in the earlier proposals and their value cannot be undervalued in any way.

Most of the key management schemes presented in this chapter use a centralized controller for performing group initialisation and rekeying tasks, which makes the controller possibly a bottleneck or a single point of failure - in theory. In practice we

can use techniques like load balancing, clustering and replication to achieve fault tolerant systems with high availability.

	GKMP	SMKD	Iolus	CKS	HTS	DISEC/OFT
Nbr of keys at the controller	n+1	2	GSI count(GSC)	n+2	$\frac{kn-1}{k-1}$	-(DISEC)/2n-1(OFT)
Nbr of keys at the member	2	2	1	n+1	$\log_d n$	$2\log_2 n$
Nbr of msgs for join	n	NA	1	n	$d\log_d n$	$\log_2 n$
Nbr of msgs for leave	n	NA	Avg. nbr. members in subgroup	1	$d\log_d n$	$\log n$
Theoretical single point of failure	Yes	Yes	Yes	Yes	Yes	No

Table 1. Key management schemes compared

The ultimate task of these schemes is to find a balance between number of stored keys at each host, number of messages to be sent during joins, leaves and group initialisations and the number of encryptions and decryptions to be performed during keying operations. The balance between these measures evaluates the scalability of a scheme. The schemes with logarithmic resource consumption growth are of course of the most desirable type at this time as shown in Table 1.

Let's take our example company Keytouch into discussion again and try to apply some of the schemes to its services. As a commercial company Keytouch has to be able to bill its customers hence a centralised controller based hierarchical key management scheme would be a viable solution for all the one-to-many type entertainment services that have to serve possibly millions of customers. For secure conferencing services we could use the DISEC with some kind of external billing and session mechanism to take some centralized control over conferencing. Or to simplify and unify the overall architecture of the service we might settle for centralised hierarchical tree solution for the conferencing applications.

6. Conclusions

In this study we have presented some mainstream proposals as how to handle key management in multicast environment. Not all the possible approaches are presented here but some of the alternatives that represent the current directions in this field and some of the schemes that have laid the cornerstones for this area of research. The research in this area has been going on for some years now and it seems that there is not yet a clear consensus on how the keys should be distributed. The main problem area in multicast key management is currently scalability. Several different proposals exist so far and the ones proposing hierarchical approach for the problem seem to have the brightest future due to their better scalability.

The multicast key management schemes presented in this study can be divided into three categories: centralized, distributed and contributory. This distinction is not strict

as some of the schemes fall into two different categories. For example DISEC is definitely a distributed scheme and its keys are formed in a contributory manner by using one-way and mixing functions. An important thing to notice is that the field of group communication is a very broad subject and that the different applications have varying scalability, performance, multicast type and security requirements. The categorisation reflects the heterogeneous requirements that are present. In this light we state that the framework for multicast applications should be open in such a way that it can host different kind of key management schemes and future enhancements.

In this study we have only scratched one corner of the group communication security and even multicasting security. Things like quality of service, session authorisation, reliable multicasting and integration with standard technologies like IPSec need also be considered. The standardisation work for multicast security issues has just started at the IETF, which indicates that the related mechanisms begin to be mature enough for broader audience.

Key management is one of the most critical issues in cryptographic systems. It forms the spine for the presence of digital security.

References

[Black, 2000] Uyless Black, *Internet Security Protocols*, Prentice-Hall, 2000.

[Cormen et al., 1990] Thomas H. Cormen, Charles E. Leiserson, Ronald R. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[Deavours et al., 1998] Cipher A. Deavours, David Kahn, Louis Kruh, Greg Mellen, Brian J. Winkel, *Selections from Cryptologia – History, People and Technology*, Artech House Publisher, 1998.

[Doneti, 2000] Lakshminath R. Doneti, Sarit Mukherjee, Ashok Samal, *DISEC: A Distributed Framework for Scalable Secure Many-to-many Communication*, Proceedings of the Fifth IEEE Symposium on Computers and Communications, 2000.

[dvmrp-v3, 2000] T. Pusateri, *Distance Vector Routing Protocol version 3*, Internet Draft, 2000.

[GKMPA, 1997] H. Harney, C. Muckenhirn, *Group Key Management Protocol (GKMP) Architecture*, RFC2094, 1997.

[GKMPS, 1997] H. Harney, C. Muckenhirn, *Group Key Management Protocol (GKMP) Specification*, RFC2093, 1997.

[IBM TCP/IP RedBook, 1998] Martin W. Murhammer, Orcun Atacan, Stefan Bretz, Larry R. Bugh, Kazunari Suzuki, David H. Wood, *TCP/IP Tutorial and Technical Overview*. International Technical Support Organization, 1998.

[Kahn, 1967] David Kahn, *The Codebreakers; The Comprehensive History of Secret Communication from Ancient Times to the Internet*, Scriber, 1967.

[Kerckhoffs, 1883] Auguste Kerckhoffs, *La cryptographie militaire*, Journal des sciences militaires, vol. IX, 1883.

[Kosiur, 1998] Dave Kosiur, *IP Multicasting: The Complete Guide to Interactive Corporate Networks*, John Wiley & Sons Inc., 1998.

[McGrew, 1998] A. McGrew, A. T. Sherman, *Key Establishmen in Large Dynamic Groups Using One-Way Function Trees,* IEEE Transactions on software engineering, 1998.

[Menezes, 1997] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[Mittra, 1997] S. Mittra, *Iolus: A Framework for Scalable Secure Multicast*, ACM SIGCOMM '97, Cannes, France, 1997.

[Moyer et al., 1999] Matthew J. Moyer, Josyula R. Rao, Pankaj Rohatgi, *A Survey of Security Issues in Multicast Communications,* IEEE Network, November/December 1999.

[Poovendran and Baras, 2001] R. Poovendran, J.S. Baras, *An Information Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes,* Technical Research Report, Journal of IEEE Transactions on Information Theory, 2001.

[RFC 792, 1981] J. Postel, *Internet Control Message Protocol*, Network Working Group, 1981.

[RFC 1075, 1988] D. Waitzman, C. Partridge, S. Deering, *Distance Vector Multicast Routing Protocol*, Network Working Group(IETF), 1989.

[RFC 1112, 1989] S. Deering, *Host Extensions for IP Multicasting*, Network Working Group (IETF), 1989.

[RFC 1583, 1994] J. Moy, *OSPF version 2*, Network working group(IETF), 1994.

[RFC 1584, 1994] J. Moy, *Multicast Extensions to OSPF*, Network Working Group (IETF), 1994.

[RFC 1949, 1996] A. Ballardie, *Scalable Multicast Key Distribution*,Network Working Group (IETF), 1996.

[RFC 2201, 1997] A. Ballardie, *Core Based Trees (CBT) Multicast Routing Architecture*, Network Working Group (IETF), 1997.

[RFC 2362, 1998] D. Estrin et al., *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*, Network Working Group (IETF), 1998.

[RFC 2463, 1998] A. Conta, S. Deering, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)*, Network Workin Group(IETF), 1995.

[RFC 2627, 1999] D. Wallner, E. Harder, R. Agee, *Key Management for multicast: Issues and Architectures,* National Security Agency (IETF), 1999.

[Taxonomy, 1999] R. Canetti, B. Pinkas, *A Taxonomy of Multicast Security Issues,* Internet Draft*,* IBM Research and the Weizmann Institute, 1999.

[ThinkQuest,1999] http://library.thinkquest.org/27993/crypto/classic/poly2.shtml

[Timeline, 1996] http://world.std.com/~cme/html/timeline.html

[Wong et al., 2000] C. K. Wong, M.Gouda, S.S. Lam, *Secure Group Communications Using Key Graphs,* IEEE/ACM Transactions on Networking, Vol.8 No.1, 2000.