A Practical Key Management and Distribution System for IPTV Conditional Access

A Dissertation
Submitted to the Department of Computer Science
Faculty of Science
at the University of Cape Town
in partial fulfilment of the requirements
for the degree of
Master of Philosophy in Computer Science
by Coursework and Dissertation

Prepared by
Gregory L. Harding
(HRDGRE001)

16 September 2013

Supervised by
Dr. Anne Kayem

Abstract

Conditional Access (CA) is widely used by pay-television operators to restrict access to content to authorised subscribers. Commercial CA solutions are available for structured broadcast and Internet Protocol Television (IPTV) environments, as well as Internet-based video-on-demand services, however these solutions are mostly proprietary, often inefficient for use on IP networks, and frequently depend on smartcards for maintaining security.

An efficient, flexible, and open conditional access system that can be implemented practically by operators with large numbers of subscribers would be beneficial to those operators and Set-Top-Box manufacturers in terms of cost savings for royalties and production costs. Furthermore, organisations such as the South African Broadcasting Corporation that are transitioning to Digital-Terrestrial-Television could use an open Conditional Access System (CAS) to restrict content to viewing within national borders and to ensure that only valid TV licence holders are able to access content.

To this end, a system was developed that draws from the area of group key management. Users are grouped according to their subscription selections and these groups are authorised for each selection's constituent services. Group keys are updated with a key-tree based approach that includes a novel method for growing full trees that outperforms the standard method. The relations that are created between key trees are used to establish a hierarchy of keys which allows flexible selection of services whilst maintaining their cryptographic protection. Conditions for security without dependence on smartcards are defined, and the system is expandable to multi-home viewing scenarios.

A prototype implementation was used to assess the proposed system. Total memory consumption of the key-server, bandwidth usage for transmission of key updates, and client processing and storage of keys were all demonstrated to be highly scalable with number of subscribers and number of services.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Thesis Statement

In the television industry, Conditional Access (CA) [8] is the application of cryptography for controlling access to content. The term CA is most commonly used in reference to the traditional scheduled programming that is distributed over digital broadcast mediums such as satellite, terrestrial, and cable.

Pay-television (or premium television) content, which consists primarily of high quality video and audio streams, is by its nature a high-bandwidth application. Traditional television broadcast networks have limited bandwidth and typically it is infeasible to encrypt the data stream with a different key for each valid subscriber. In recent years, increasing availability of fast home broadband connections has allowed the advent of Internet Video-On-Demand (VOD) services such as Netflix[1] which do establish individual network connections (unicast) to clients. This type of application is reportedly responsible for a large percentage of global bandwidth usage, placing a strain on distribution networks. Unicast distribution of "live television" (including scheduled programming) to a large number of receivers is extremely inefficient. Multicast facilitates efficient distribution of this type of content over IP networks [9].

Pay-TV operators are heavily reliant on conditional access systems to support their business model. Conditional Access Systems (CASs) are also used for Free-To-View (FTV) television broadcasts. FTV is distinct from Free-To-Air (FTA) systems in which the broadcast content is not encrypted at all. Encryption of FTV content

[1] `http://www.netflix.com/`

can be for reasons such as restricting access to content by physical region. National borders, for example — to prevent viewers in nearby countries from accessing content that is ultimately paid for by tax-payers. Within a country, regional boundaries might also be defined on the basis of broadcast licenses and advertising.

In the near future, the South African Broadcasting Corporation (SABC) will be following in the footsteps of its counterparts around the world (including the United States of America, United Kingdom, and Australia) in switching over[2] to Digital Terrestrial Television (DTT). The benefits of digital television include greater efficiency in usage of the radio spectrum, and better picture quality. The SABC could use a CA system to limit access to within South Africa, or for ensuring that only valid TV-licence holders are able to watch TV.

1.2 Motivation and Objectives

This research was inspired in part by the need of a South African Set-Top-Box (STB) manufacturer for an open, royalty-free conditional access mechanism for Internet Protocol Television (IPTV) [16] that is based on results from research in cryptographic key management. The objective of this research therefore is to study existing research in key management systems and design and implement an end-to-end framework for conditional access that is both theoretically robust and practically implementable.

Existing commercial CA systems are closed, and proprietary. To the best of our knowledge, while there is much research in the area of group key management, there is little available research that specifically addresses the design of a complete conditional access system for IPTV.

Furthermore, some existing conditional access systems rely on "security by obscurity". That is, the strength of their security is dependant on an attacker not having knowledge of the workings of the system. For example, a subscriber's viewing selection might be enforced by software disallowing certain services. A restriction enforced in this way can be bypassed by an attacker who implements their own receiver system. *Kerckhoffs' principle* [23] should be applied so that an attacker having knowledge of the algorithms used should not compromise the security of the system. Therefore there is arguably room for improvement in terms of security of the existing approaches.

[2]SABC DTT migration: `http://www.sabc.co.za/wps/portal/SABC/dtt`

CA systems used in some IPTV deployments are based on Digital Rights Management (DRM) mechanisms [4], use a simple key distribution centre (SKDC) approach, or encapsulate broadcast CA systems in IP packets. There is room for improvement in terms of efficiency relative to these approaches by specifically tailoring a solution to the IP multicast distribution mechanism.

1.3 Contributions

The core achievement of this research is the design of a framework that can be used as the basis for a conditional access system. In order to address this larger objective of how to build a conditional access system, the following research questions were posed.

How best can existing research in key management techniques be combined to provide a flexible and practical conditional access system for IPTV?

This research considers the scalability and efficiency problems that are encountered when attempting to securely deliver content to a large group of authorised subscribers. A survey and assessment is made of techniques in group key refreshment and distribution, reliable key delivery, and secure storage of keys.

The proposed conditional access system uses an efficient key-tree based method for sharing a group key and associated updates with a large group of subscribers. This method attempts to minimize overhead related to key management in terms of processing, memory usage, and bandwidth. The CA system includes a fast technique, which we believe to be novel in this application, for growing full key trees that reduces the amount of data to be sent for a key update.

Multiple services are supported by using a hierarchical key structure for grouping subscribers according to their viewing selections. This structure defines a relationship between key trees that enables flexible viewing selections to be cryptographically enforced. In order to improve upon the poor scalability of existing techniques, service groupings known as bouquets are incorporated to accommodate a large number of services.

Can an adaptable conditional access solution be found that is secure without the need for smartcards?

Most existing television delivery environments rely on a smartcard being held by each subscriber. The smartcard is used as an identity module and as a tamper-resistant device for storing cryptographic keys. This dependence restricts the type of devices to which content can be delivered. To support devices such as mobile phones and tablets, it is necessary to negotiate and store keys securely without the need for a smartcard.

The proposed system specifies a protocol for sharing a secret key between a key-server and a receiver device. Furthermore, requirements for the secure storage of keys in the non-volatile memory of the receiver device are defined. These requirements assume the availability of a trusted platform.

Can the solution be extended to allow sharing of media content in an authorised domain?

Increasingly, users want the flexibility to view content on multiple devices throughout their households. We propose an extension to the aforementioned service grouping structure that supports multiple viewing devices attached to a single subscription without compromising cryptographic strength of the system.

1.4 Notation and Terminology

The following notation indicates transmission of a message m by server s to user u, encrypted with key k:

$$s \rightarrow u : \langle m \rangle_k$$

The notation is extended to indicate transmission to multiple users and a message with multiple components. The message might be broadcast or multicast. The list of users is those for whom it is intended, although due to the nature of these routing mechanisms, there is no guarantee that they receive it, or that others do not. Security is therefore ensured by the use of encryption:

$$s \rightarrow u_1, u_2 : \langle m_1, m_2 \rangle_k$$

Throughout this thesis, the term *subscriber* will be used interchangeably with *user*, *client*, and *receiver*. Unless otherwise specified this indicates a receiver of content that is transmitted by a server and is not intended to imply any specific business relationship (eg. paid subscription) unless otherwise indicated. *Client* is used in the context of a client/server relationship.

The term *server* will primarily refer to a key management and distribution server as part of the complete head-end. *Head-end* refers to the operator's equipment that is used for processing and delivering a television broadcast. This includes encrypting elementary audio and video streams, multiplexing them and including other necessary data, and uplinking to satellite.

In digital television, *service* is used to refer to a particular stream of content programming. In common usage this is referred to as a television "channel". Channel more properly refers to the particular means of transmission – for example a radio frequency.

Operators often define *bouquets* which are logical groupings of services. These are most often used as atomic subscription units — that is, groupings such as "Movies", "Sport", or "Documentaries". These groupings are not necessarily non-overlapping: a service might be in multiple bouquets, thus allows groupings such as "Compact" and "Premium".

Where set notation is used, $A \cap B$ means the intersection of sets A and B. An element is contained in $A \cap B$ if and only if it is contained in set A and set B. $A \subseteq B$ means that A is a possibly-equal subset of B. $A \setminus B$ means the set of elements that are present in set A but not in set B.

1.5 Outline

This section provides a brief outline of the remainder of this thesis.

Chapter 2 covers general background information on television distribution, and digital pay television in particular. An overview of networking, cryptography, and secure communications is provided.

In Chapter 3 a detailed assessment of cryptographic group key management and access control techniques is undertaken with consideration for the specific requirements of a conditional access system for IPTV.

Chapter 4 outlines the design of the proposed conditional access system for this research. The results obtained with a prototype implementation of the system are presented and evaluated in Chapter 5, and finally the conclusion in Chapter 6 discusses the outcomes of this research and highlights areas for further work.

Chapter 2

Background

2.1 Television Distribution

Television is an integral part of the fabric of modern society. It has been widely available for the better part of a century, and has become established as a primary medium for the distribution of news and entertainment. It provides a means for organisations such as governments and corporations to reach large audiences efficiently and effectively. The role television has played in forming public opinion since the mid-twentieth century cannot be understated. It is a technology that has truly made the world a smaller place.

Historically, television has been *broadcast* to viewers by means of terrestrial radio transmissions and cable networks. The advent of telecommunications satellites in the 1960s and 70s saw the introduction of "direct-to-home" satellite television broadcasts. Satellite television is highly scalable as it does not require the installation and maintenance of complex infrastructure to support additional viewers. Satellite broadcasts can cover vast areas that would be impractical or uneconomical to service with cable or terrestrial broadcasts. Viewers only need to be supplied with low-cost receiving equipment: a STB attached to a parabolic satellite dish.

2.2 Internet Protocol Television

Internet Protocol Television (IPTV) refers to the delivery of television content over an IP network. It is distinct from video content provided by Internet services such

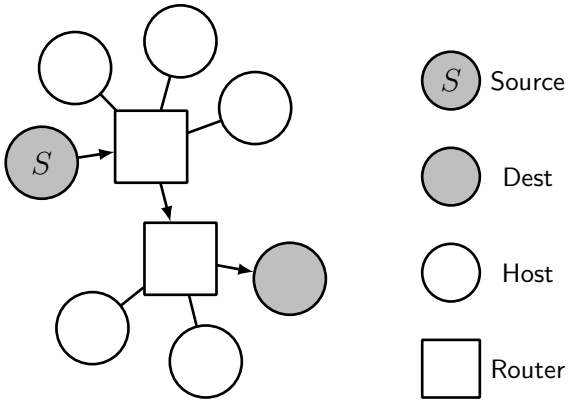
as Youtube, Netflix, and Hulu (although the security principles described in this thesis could be applied to some of these services). Montpetit et al. [27] provide a comprehensive overview of the current technologies, practices, and trends involved in the delivery of IPTV content. Their overview includes a discussion of IP multicast and related transport protocols.

2.2.1 Multicast Networking

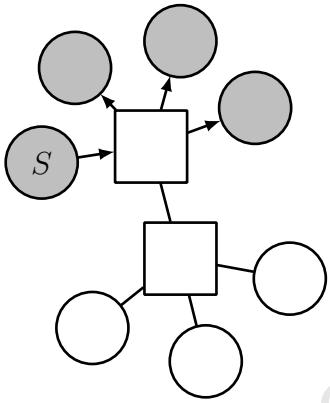
An IP network is packet-switched. This means that messages are broken up into *packets* and routed from a source to a destination either directly on a local network segment, or via intermediate routers. The three primary schemes for routing these packets are *unicast*, *broadcast*, and *multicast*. Unicast refers to a packet being sent to a single destination. Figure 2.1a shows an example where sender S transmits a single packet that is intended for the single destination (highlighted). The packet is routed across network segments to reach the destination. Broadcast involves simultaneous delivery to all destinations on a local network segment. The example in Figure 2.1b shows S sending a single packet that is delivered to all destinations on the same network segment as S, whether they want it or not. Multicast involves simultaneous delivery of the packet to a particular set of destinations who have registered an interest in receiving them by joining a particular multicast group. In Figure 2.1c the single packet that is transmitted by S is received by those destinations that have told their closest router about their interest in it.

The important factor is that broadcast, and particularly multicast, are features implemented by the network infrastructure. The source host transmits a single packet that is delivered to multiple destinations by the network switching and routing equipment. This significantly reduces the load on the source host and the network that would otherwise be required to transmit the packet separately to each destination by the successive or simultaneous establishment of multiple unicast connections [34].

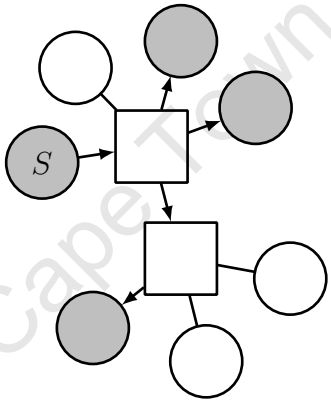
Broadcast is a feature implemented in IPv4 but not available in IPv6 [11], and is primarily restricted to local networks and associated technologies. It makes use of a specific broadcast address. Packets sent to this address will be received by all hosts on a particular network segment. Broadcast traffic is not routed outside of Local Area Network (LAN) segments because it causes tremendous load on receiving hosts and the network, even if they are not interested in receiving the packets.



(a) Unicast



(b) Broadcast



(c) Multicast

Figure 2.1: Packet routing strategies. With unicast, a packet is routed from a source to a single destination. With broadcast, a packet is simultaneously delivered to all hosts on the same network segment. With multicast, a packet is simultaneously delivered to all interested hosts.

	Broadcast	Unicast	Multicast	Return Channel
Satellite	✓			✗
Terrestrial	✓			✗
Cable	✓			✓(some)
IPTV		✓	✓	✓

Table 2.1: Comparison of Television Transmission Methods. This table highlights the *in-band* features of common transmission methods. Specifically, *out-of-band* return channels via Internet connections, for example, are not considered as they require completely separate transmission systems.

We distinguish between IP broadcast/multicast and television broadcasting by satellite or terrestrial radio signals. Conceptually, broadcast of information by radio signals to all listening receivers is similar. However it is a highly efficient and scalable use of radio technologies to distribute from a single transmitting source to multiple receivers, all of which are interested in receiving the signal (assuming they are powered-on and tuned to the correct frequency).

An IP network has broader, more general applications than over-the-air television networks. Typically it is not dedicated to television delivery and the assumption cannot be made that all hosts want to receive all packets. IP multicast facilitates efficient use of the network infrastructure.

Clients indicate the groups they are interested in by communicating with their nearest router using the Internet Group Management Protocol (IGMP) [9] on IPv4 networks and Multicast Listener Discovery (MLD) [10] on IPv6 networks. This router in turn communicates with other routers, and so they build up routing tables that ensure only relevant traffic passes through them. Groups are identified by specific IP addresses. The range of IPv4 addresses that is allocated to multicast is from `224.0.0.0` to `239.255.255.255`. In IPv6 the `ff00::/8` address block is used.

Data is usually sent to a multicast group using User Datagram Protocol (UDP) datagrams. Transmission Control Protocol (TCP) is not applicable to multicast traffic as it only supports two-way, one-to-one connections as opposed to UDP which allows for point to multipoint transmission.

The Real-time Transport Protocol (RTP) is commonly used [27], encapsulated in UDP datagrams, for delivering streaming media content. It defines various *profiles* for different types of content. RTP Control Protocol (RTCP) is a complementary protocol primarily used by receiving clients to communicate statistics back to the server that it can use to tune its output (such as quality-of-service and rate limiting parameters). Secure RTP (SRTP) [5] is a particular profile defined by RTP for encryption, integrity verification, and message authentication. It uses Advanced Encryption Standard (AES) with a counter stream cipher mode of operation.

2.2.2 Return Path

Satellite and terrestrial television distribution networks offer purely one-way data transmission. That is, they do not provide any *in-band* mechanism for return communication from the viewer to the operator (head-end).

A *return-path* (or *return-channel*) is a communications channel that the receiver may use to transmit data to the head-end. The lack of a return path places certain limitations on the type of services that can be offered on these networks (for example, proper video-on-demand, which is described further in Section 2.3.3). The routing strategies and return path features offered by the various television transmission methods are summarised in Table 2.1.

As the scope of this research is directed at IPTV, it primarily considers the situation where a permanent return-path is available. This allows for the assumption to be made that a reliable unicast fallback mechanism is available for a client to request keys. Apart from IPTV, modern cable television deployments include two-way communications, and satellite and terrestrial broadcast systems can be augmented with separate return channels (for example, a satellite-IP *hybrid* set-top-box). Such a hybrid system is useful for countries such as South Africa, where broadband Internet access is currently limited in terms of bandwidth and data usage caps.

2.3 Pay Television

The primary application of conditional access is to protect the delivery networks of pay-television operators. An operator may offer content and services to customers in various ways including subscriptions, "pay-per-view", and video-on-demand. The Conditional Access system that they use must be able to accommodate these different options.

2.3.1 Subscription

The pay television industry operates a predominantly subscription based business model. They are funded partly or fully by paying subscribers, as opposed to advertising revenue, or public-funding. This allows a broader selection of service content types to cater to viewers' interests. Subscriptions allow access to scheduled programming that is available from a selection of services.

2.3.2 Pay-per-view

Pay-Per-View (PPV) is a service which might be offered by pay television operators to existing subscribers or once-off customers who have the necessary equipment

to receive the transmission. All viewers tune-in at a scheduled time to receive the broadcast, which is typically of a special event or sports match. PPV can be thought of as a very short-term subscription.

2.3.3 Video-on-demand

Video-on-demand (VOD) is a service that allows viewers to individually choose content from a selection and watch it at their leisure. This is achieved in different ways, depending on the transmission mechanism being used. In a pure broadcast environment, Push-Video-On-Demand (PVOD) requires a subscriber to have a device with a local storage capability such as a Digital Video Recorder (DVR) with a hard disk. A selection of content is background downloaded from a dedicated broadcast channel and saved to the disk. Once some content is available on the disk, the subscriber is able to view it at their convenience. With "true" VOD, content is streamed directly from a server via a unicast connection.

2.4 Content Protection

In many forms of communications all parties share at least some interest in maintaining the security of said communications. Content protection, however, is primarily aimed at protecting the interests of service operators and media copyright holders. Thus there is little incentive for customers to take steps to maintain security, and when these steps are enforced on legitimate customers they are seen as an unwelcome hindrance (particularly in the case of DRM applied to media such as music and books — i.e. outside the confines of a controlled distribution network). Additionally, the trend is towards consumers being able to access media across a variety of devices at any time — for example, a consumer might wish to watch a show on a television in the comfort of their lounge, or on a mobile device when travelling. Content protection mechanisms should avoid preventing consumers from having the convenient access to media that they desire, or the consumers might resort to illegitimate methods of obtaining the media that enable to them consume it freely.

It is both pragmatic and prudent for operators and vendors of content protection systems to work on the assumption that the system can never be completely secure. Given enough time and resources, any system can be broken.

The goal therefore is to ensure that the cost of breaking the system is more expensive than using it legitimately would be. It should be more costly and/or onerous for potential customers to attempt to avoid paying for a subscription than to simply pay. Particularly, it should not be possible for illegitimate efforts to piggyback on the legitimate network. For example, an operation that attempts a pirate redistribution of the service for commercial gain should not be able to make use of the operator's distribution network. This is a problem with over-the-air transmissions via terrestrial and satellite where cloning smartcards or sharing decryption keys obtained from legitimate subscriptions are often the only steps necessary to set up an illegal commercial operation [28, 47].

If some part of the content distribution network is vulnerable, the strength of other areas is irrelevant. The primary weakness in any such system is at the point where, by necessity, the content must be decrypted in order for customers to view it. This is addressed by employing an end-to-end chain of trust. The data is transmitted securely by the CA schemes discussed in this thesis. The receiving device (STB) then sends it to a television via an encrypted signal using the HDCP (High-bandwidth Digital Content Protection) protocol, where the source device cryptographically authenticates the destination device before sending the data. The destination (television) displays the content. In theory the only way to extract the content would be to record a lower quality version via analogue device outputs. The high quality digital data is encrypted from its source at the operator all the way to the TV.

2.4.1 Conditional Access

In order to enforce their business model, pay TV operators need some way of restricting access to their content to only paid-up subscribers. In early cable TV systems, access could be restricted by using signal filters at the cable taps for each subscribers' premises to remove certain channels from their signal. This method was not scalable with an increase in the number of channels and subscribers (due to the large number of signal filters and locations involved) and it was relatively easy to work around the filters illegitimately. Alternative methods such as distorting or scrambling the signal with interference signals were used, but these could also be bypassed. The advent of digital television allowed solutions using encryption and associated key management schemes to be applied to the problem. These cryptographic approaches are at the core of current conditional access solutions.

2.4.2 Digital Rights Management

Digital rights management (DRM) and conditional access aim to achieve similar goals: namely the protection of content that is ultimately distributed to untrusted users. DRM is a widely applicable term that refers to any mechanism for restricting the use of content or hardware.

Azad et al. [4] provide an overview of research in DRM. Broadly, DRM associates content with a set of *usage rights* and protects both the content and the rights, usually by cryptographic means. It attempts to associate a unique copy of some content with a specific authorised user. Each authorised user would receive their own copy of the content. This is done in such a way that a user should only be able to access his/her own copy of the content, and then only in the manner defined by the provider of the content. For DRM to be effective, only trusted hardware or software should be able to decrypt and interpret the rights, and provide a user with access to the content only if the rights and additional authorisation information permit doing so.

DRM aims to restrict the use of content that is distributed or distributable outside the confines of a controlled system (i.e. an untrusted distribution channel). Conditional access is part of a more tightly controllable system and aims to protect the distribution network itself. It is distinct from DRM in that it attempts to protect a single copy of content that is shared by all users [27].

2.5 Digital Video Broadcasting

The Digital Video Broadcasting (DVB) Project[1] defines a comprehensive set of open standards that are widely used in Europe, Australia, Africa and other parts of the world for the broadcast of digital television[2]. They specify most aspects of digital television to allow interoperability and compatibility between networks and equipment. For example, the DVB-S standard [14] specifies modulation, encoding and transmission characteristics, and data formats and protocols for satellite television. The newer DVB-IPTV [16] includes specifications for the use of protocols such as RTSP and RTP, and multicast networking.

[1] `http://www.dvb.org`

[2] Whilst DVB is used widely, similar competing standards are in use elsewhere: the Advanced Television Systems Committee (ATSC) standards in North America; and the Integrated Services Digital Broadcasting (ISDB) standards in parts of Asia and South America.

The standards also define the DVB Conditional Access framework (DVB-CA) [7, 8] that allows operators to use various commercial conditional access solutions. Major vendors of such solutions include Irdeto[3], NDS[4] and Nagravision[5]. These systems are typically closed, and proprietary, with almost no information publicly available on the exact mechanisms that are used to achieve the required objectives.

In DVB-CA video is scrambled with a *control word* (CW) which is used as a seed to a pseudorandom number generator. Either the common scrambling algorithm (DVB-CSA [17]) or optionally some variant of it is used, as decided by the vendor of the particular conditional access system. The subscribers' receiving equipment needs a compatible descrambler.

The CW is common for all subscribers of the system and is changed often (every few seconds) to minimize the impact of key discovery. A relatively small key size is used to allow real-time descrambling on relatively constrained embedded systems. This key is potentially vulnerable to brute force and side-channel attacks [24, 44]. Newer versions of DVB-CSA which take faster hardware into account provide stronger AES-based encryption.

The DVB-CA standard specifies the existence of *Entitlement Control Messages* (ECMs), which contain CW updates, and *Entitlement Management Messages* (EMMs) which contain other instructions and key updates for the CA system. These messages are opaque, and usually encrypted, with their format and content specific to the particular conditional access system in use. They are decrypted and processed by the CA system, which must ensure that it provides a control word to the descrambler in a timely fashion.

Any mechanism that fits into this specification can be used to securely deliver the CWs. In the most simplistic system, the CA system might store a static, globally shared master key in smartcards (which are tamper resistant). This key would use symmetric cryptography to decrypt the control words. The impact of this single key being compromised is severe because it could be widely distributed, thereby resulting in a complete break of the system.

More complex systems will encrypt the CW with an intermediate key-encryption-key (KEK). The intermediate KEK is usually stored on authorised smartcards and periodically updated using a message encrypted with a smartcard's public key. One such

[3]Irdeto B.V. `http://www.irdeto.com/`
[4]NDS Limited. `http://www.nds.com/`
[5]Nagravision SA. `http://www.nagravision.com/`

message would need to be delivered per smartcard in circulation. The performance and bandwidth requirements scale linearly $O(n)$ with this approach, sometimes referred to as as Simple Key Distribution Centre. Even more advanced systems employ hierarchies of keys that allow securely delivering these keys in a more efficient manner.

It should be noted that the use of a control word allows flexibility in the system: it can be updated very frequently without incurring further rekey operations; it facilitates the "modular" interchange of CA systems; and in fact it allows multiple CA systems to be used simultaneously, with an ECM for each system containing the same control word (this is known as *SimulCrypt* [15]).

2.6 Secure Network Communications

This section provides a general overview of the principles of secure network communications between a sender and a receiver. The same principles can be applied to multicast communications and specifically for the pay-TV scenario where there is a single sender but multiple receivers.

2.6.1 Attacks and Vulnerabilities

Before attempting to secure a system, it is necessary to have an understanding of the types of threats to which the system could possibly be subjected. The International Telecommunication Union (ITU) X.800 Recommendation [21] defines a security architecture that specifies categories of *security services* and classification of *attacks* on the security of a system.

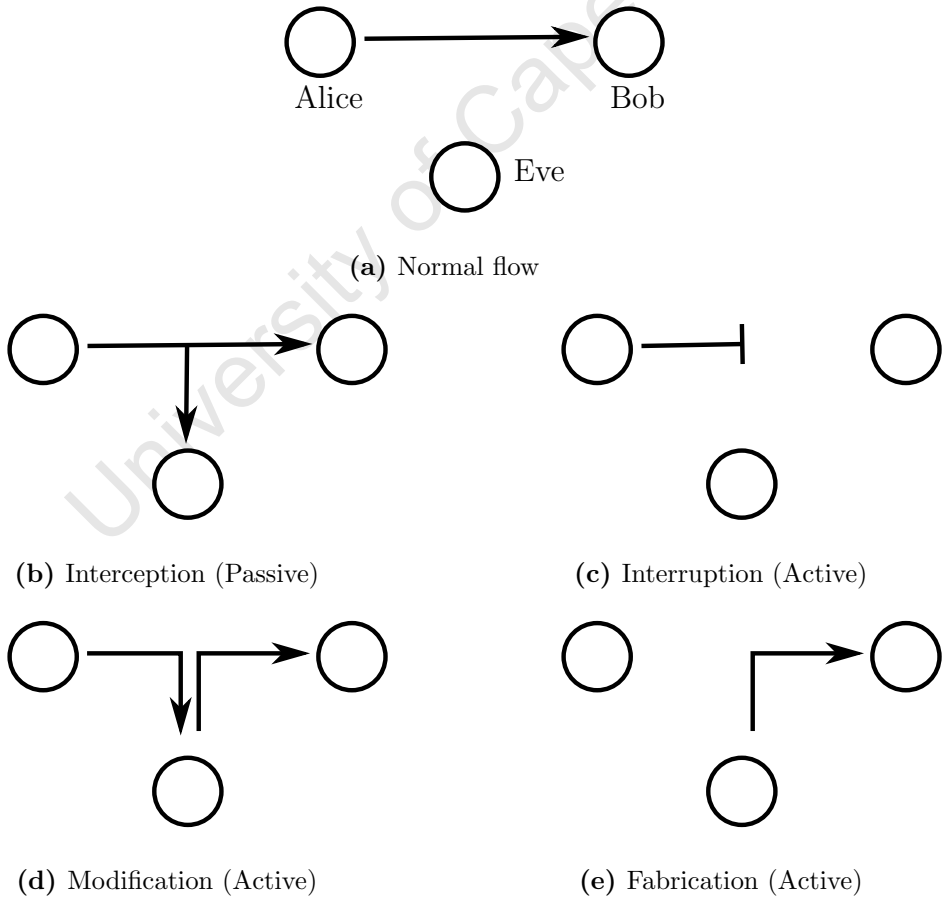
In normal communications, a message is passed from a sender to a receiver. This is illustrated in Figure 2.2a where Alice sends a message to Bob. It may pass through multiple intermediaries (network routers) along the way. Attacks by a third party, Eve, on this data flow can be classified as either active or passive. In a passive attack, the message arrives at the receiver unmodified from the form in which it left the sender. In an active attack the message that is received is altered, fabricated, or not received at all.

Interception is a passive attack on data confidentiality. Figure 2.2b shows Eve is able to read a copy of the message that is intended only for Bob. This is done without affecting Bob's receipt of the message.

Interruption is an active attack on availability. Figure 2.2c shows that some action taken by Eve prevents Bob from ever receiving the message sent by Alice.

Modification is an active attack on data integrity. In Figure 2.2d, Eve is able to receive the message that is sent by Alice and modify it before sending it on to Bob. Bob is unaware that the message he receives is changed from the one sent by Alice.

Fabrication is an active attack on authenticity. Eve is able to create a message that appears to originate from Alice, and send it to Bob. This is shown in Figure 2.2e.



(a) Normal flow

(b) Interception (Passive)

(c) Interruption (Active)

(d) Modification (Active)

(e) Fabrication (Active)

Figure 2.2: Possible forms of attack on a message sent from Alice to Bob. Eve is the attacker.

2.6.2 Security Objectives and Services

Information security has certain objectives that need to be met dependant on the requirements of the particular application. These desirable properties known as *security services* are implemented by various underlying mechanisms and protocols [40].

Confidentiality

Confidentiality ensures that information is only disclosed to authorised recipients. An attacker eavesdropping on a network will not be able to read the information. It is protection against interception attacks.

In the context of pay-TV, confidentiality is required to ensure that only authorised subscribers are able to access content.

Integrity

Integrity is the ability to ensure that the data received is unchanged from what was sent. It provides protection against modification attacks and against accidental corruption of data that might occur in transit.

Availability

A system must remain available to those for whom it is intended. Denial-of-service and interruption are attacks against availability.

Authentication

Authentication is the process of one entity verifying the identity of another — checking they are who they claim to be. Authentication might be conducted mutually or in one direction only. It protects against fabrication attacks.

For example, the pay-TV operator needs to authenticate subscribers (specifically their receiving device) to associate them with a valid subscription.

Access Control

Access control restricts who can join a network or make use of a service. Once a user is authenticated, it grants them access in accordance with their associated authorisations.

Non-Repudiation

Non-repudiation is the ability of a system to ensure that a user cannot deny involvement in communications that have been sent or received and acknowledged.

In the context of business systems, such a service would be required to prove that a customer placed an order, for example.

2.7 Cryptography

Provision of the aforementioned security services is primarily accomplished by performing certain transformations to the data before sending it, and again once it has been received. These transformations make use of *cryptography* and are usually achieved with the use of *keys*. The security of a system depends on the restricted availability and careful distribution of these keys. For example, a message is encrypted with a secret key to provide confidentiality and only parties who hold the appropriate secret key for decryption will be able to understand the message. This management and distribution of cryptographic keys is a core function of a conditional access system.

2.7.1 Symmetric Cryptography

With symmetric cryptography, the same secret key is used to decrypt the data as was used to encrypt it [36]. Block ciphers such as DES (Data Encryption Standard) [1] and AES (Advanced Encryption Standard) [2] are examples of symmetric cryptography.

The primary issue with symmetric cryptography is finding a way to securely distribute the secret key to all concerned parties. If the secret key is intercepted, the encryption is worthless.

Various methods of secure key distribution exist. A key might be pre-shared between a group of users (for example, by physically entering the same password into multiple computers, or loaded onto a STB during production in the factory). Alternatively some key exchange protocol might be used. For example, the Diffie-Hellman key establishment protocol [12] allows two parties to securely negotiate a symmetric key over an insecure channel. Even if all of the communication is intercepted, an eavesdropper will not be able to determine the negotiated key.

2.7.2 Asymmetric Cryptography

Asymmetric cryptography (also known as public/private key cryptography) uses a separate key for the encryption and decryption steps [20]. Encryption is typically performed with a public key which can be widely distributed without concern for security. Only the holder of the private key is able to decrypt the message. RSA [35] is a widely used example of an asymmetric cryptographic algorithm.

Asymmetric cryptography is significantly slower than symmetric and is only suitable for encrypting small amounts of data. Often communications protocols might start with asymmetric cryptography to share a symmetric "session key" which is then used for the remainder of the communications. An initial exchange of public keys allows the parties to encrypt messages that can only be read by the other.

Assume E is an encryption function, and D a decryption function. In practice these functions might be the same, depending on the encryption algorithm. Given a keypair $\{K_r, K_u\}$ where K_r is the private key and K_u is the associated public key, a message m can be encrypted to produce ciphertext $c = E(K_u, m)$. Decryption with the public key is as follows $m = D(K_r, c)$.

Asymmetric cryptography can also be used to sign messages to achieve authentication. A message m is signed by generating a hash or checksum h with a hash-function H (eg. MD5, SHA). h is then encrypted with the private key K_r to produce a signature $s = E(K_r, s)$. Anyone holding the public key K_u can decrypt the signature to obtain h and then apply H to the message m_1 that they received to obtain h_1. If $h = h_1$ then the receiver is assured that the message has not been altered since it was created by the sender. Only the sender who holds the private key can generate signatures that will authenticate in this way.

2.7.3 Trusted Third Party to Establish Authentication

Unless two parties have communicated in some way and shared some secret information known only to each of them, it is usually necessary to make use of a trusted third-party for authentication (that is, for one or both of the parties to verify that the other is who it claims to be).

This service is normally provided by a certification authority [22]. Such an authority will cryptographically sign a certificate containing the public key of a party whose identity they can verify in some way. That certificate can then be used in future exchanges with unknown entities, who are able to verify it using the public key of the certification authority.

Authorities can verify other authorities, who in turn verify users. If a user trusts a particular authority, then he implicitly trusts another authority trusted by the first, and so on. A Public-Key Infrastructure (PKI), which comprises the mechanisms and systems necessary to enable the use of digital certificates, provides this *chain-of-trust*.

2.8 Summary

In this chapter, we have presented an overview of the technologies involved in digital television transmission and introduced the need by operators to protect this digital content. In Section 2.1 we discussed traditional television distribution via terrestrial, cable, and satellite broadcasting. In Section 2.2, IPTV was introduced as a versatile medium for distributing media content to a variety of devices. Multicast, the networking routing mechanism that enable scalable IPTV implementations, was discussed in Section 2.2.1 and compared to the traditional methods of broadcasting. In Sections 2.3 and 2.4, the particular use cases of pay-television and the need for content protection were outlined. The Digital Video Broadcasting (DVB) Standards and the structure they define for supporting Conditional Access schemes was described in Section 2.5. Finally, in Sections 2.6 and 2.7, the principles for secure communications over an untrusted network were introduced as a set of distinct security services that protect against particular categories of threats. These services can be implemented by the application of cryptography.

Chapter 3

Group Key Management

3.1 Conditional Access Requirements

Primarily, a conditional access system needs to ensure that only legitimate users are able to access content, and that they are only able to access the content they are authorised to receive. That is, it must protect content from outsiders who have no subscription, and it must protect against subscribers being able to illegitimately elevate their access privileges. A conditional access system must reasonably support an operator with tens to hundreds of services, and millions of subscribers. The system needs to scale well in these ranges.

We might assume[1] that a single high-definition (HD) television service requires an average 12 Mbit/s for a suitable quality video stream. An associated 6-channel surround sound audio stream might have a bitrate of 320 kbit/s. A satellite television transponder with a maximum effective bandwidth of 64 Mbit/s would therefore be able to transmit a maximum of 5 such HD television services. Transponders are costly and steps are taken to maximise the number of services that can be delivered on one. Although other transmission mediums such as cable or IP networks have higher available bandwidth, it is not unlimited. In IP networks, bandwidth usage costs are often the burden of the end-user. A CA system should therefore attempt to minimise the data transfer overhead that it imposes.

With these requirements in mind, and to provide the security services that were outlined in Section 2.6.2, a method of sharing keys amongst a large group of users is

[1] Assumptions are based on the use of MPEG-4 AVC/H.264 video compression and AC-3 audio compression: `http://www.avchd-info.org/format/`

required. Group key management is the area of research that addresses the problems of key management and secure distribution that were highlighted in Section 2.7 with respect to a group of users.

The goal is to securely share appropriate keys amongst the authorised subscribers so that they are able to decrypt the content. The mechanism by which the keys are shared needs to be secure against collusion between users, and it must provide forward and backward secrecy. Forward secrecy means that when a subscriber is removed, they are not able to access content that is transmitted in future [33]. Backward secrecy means a new subscriber will not be able to access content that was transmitted prior to their joining. For example, an unauthorised user might capture encrypted transmissions for some period of time (without holding a subscription). He could then take out a subscription with the sole intention of obtaining decryption keys and then cancel the subscription immediately. Without backward secrecy, he could then decrypt all the previously captured content, without having paid for it.

In Section 3.2, the concept of a hierarchy of keys is introduced as the basis for encryption of video content. Various different methods for efficiently updating a shared group key are surveyed and assessed in Section 3.3. The problem of securing multiple services so that subscribers only have access to what they have paid for is addressed in Section 3.4. Reliable delivery of key update messages, and fallback mechanisms for clients to recover from missed keys are discussed in Section 3.5. Significant reductions in bandwidth usage can be had from updating keys in batches. An overview is given in Section 3.6. Finally, solutions to the problem of secure key storage without a smartcard are considered in Section 3.7.

3.2 Secure Broadcasting

In the pay-television scenario, a single source (the head-end server) needs to transmit a large amount of data simultaneously to a large number of receivers (the subscribers). The data transmission is continuous, which is known as *streaming*, and consists of video and audio streams, and other supporting information such as an *electronic program guide* (EPG). It is measured in terms of bandwidth requirements (transfer rate), rather than total data size.

In a typical over-the-air (OTA) broadcast network (such as satellite and terrestrial television), it is necessary to encrypt this data because it is otherwise impossible to control access to it. Anyone who can acquire or fabricate the necessary reception

equipment could intercept the transmission [28, 47]. On a cable or IP network, although the subscriber's connection and access is controllable by the network operator, encryption is also highly desirable to protect against interception attacks. Such attacks are very possible over large public networks such as the Internet or by means of illegal taps into private cable networks. Such taps are difficult to detect due to the networks' size.

It should be clear that attempting to secure this data transmission using the previously discussed mechanisms for two-party communications would require a separate encrypted video stream for each individual subscriber. So while that would meet the security requirements for controlling subscriber access, it is an extremely inefficient use of limited and costly bandwidth.

3.2.1 Hierarchy of Keys

An efficient solution to securely transmitting the video data stream is to encrypt it only once and then send it to all subscribers. It is necessary to securely share the key used to encrypt the video stream with all subscribers. The data encryption key can itself be encrypted with a private "key-encryption-key" (KEK). If a key k_i, is associated with each subscriber u_i, and known only to that subscriber and the server, then the data encryption key can be securely transmitted to all subscribers. The data encryption key that is shared amongst the entire group of subscribers is called a *group key* (GK). The messages that the key server would send to distribute a message and the associated group key are shown in Figure 3.1. This 2-level key hierarchy is shown in Figure 3.2.

$$
\begin{aligned}
s \rightarrow u_1, u_2, \ldots, u_n &: \langle m \rangle_{GK} \\
s \rightarrow u_1 &: \langle GK \rangle_{k_1} \\
&\vdots \\
s \rightarrow u_n &: \langle GK \rangle_{k_n}
\end{aligned}
$$

Figure 3.1: Basic protocol for distribution of a message m, encrypted with group key GK to users $u_1 \ldots u_n$. For each i such that $1 \leq i \leq n$, GK is encrypted with the private key k_i known only to s and user u_i.

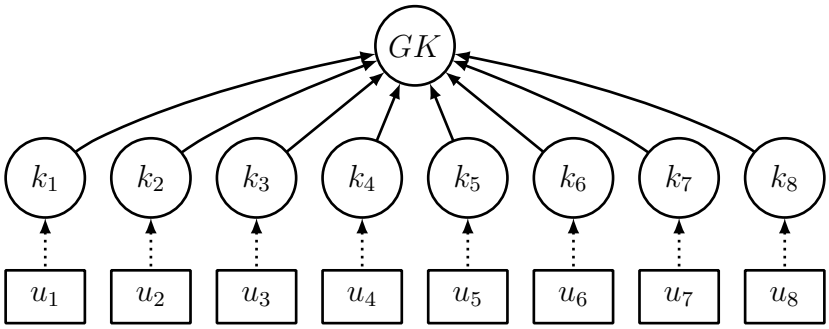


Figure 3.2: Illustration of a 2-level key structure

3.2.2 DVB Key Hierarchy

In the DVB-CA system, the actual key used to encrypt the streamed data is known as a Control Word (CW) (see Section 2.5 for more details about the DVB standards). The group key could be used as the CW directly, provided that it meets the standard format for DVB descrambler equipment. However, in the interests of flexibility it makes sense to introduce a third level in the key hierarchy. The CA mechanism is therefore not limited by descrambler hardware specifications that use a small key size. Therefore the control word is encrypted with the group key. For the purpose of this thesis, it is assumed that the actual data to be encrypted by a conditional access system is a continuous series of control words.

3.3 Group Rekeying

The group key needs to be changed every time a new subscriber is added (to prevent them being able to decrypt any transmissions made prior to their joining) or an old subscriber is removed (to prevent them being able to decrypt any future transmissions). This process is referred to as *rekeying*. In a network with a large number of users, and frequent join and remove operations taking place, the overhead in terms of bandwidth and server load for these key distributions can be significant if the simple protocol described above is used.

For example, assume a network has 10 million users and the size of a rekey message is 54 bytes (see Section 4.7 for the format of a suitable rekey message). Then for every join or leave operation the server needs to perform 10 million encryption operations, and needs to transmit approximately 515 MB of incompressible data. If the data is broadcast or multicast, then every client needs to receive all the data to find the

one key message applicable to them. If the data is unicast then the load on a client is negligible, however the server needs to make 10 million outgoing connections — which places high demands on a system that can multicast all other data (video and audio).

This section discusses a number of alternative methods for performing these group rekey operations. The methods are compared and assessed with regards to their applicability to a conditional access system. The comparison is given in Section 3.3.7.

3.3.1 Stateful and Stateless

Zhu and Jajodia [48] classify methods for group rekey operations as either *stateful* or *stateless*. This state refers to the information a client needs to hold in order to receive and interpret a new key update message.

In a stateful system, a client needs to have successfully interpreted key update messages that were previously distributed and must hold the keys contained in them in order for it to be able to decrypt new update messages. As an example, consider the situation in which each new group key is encrypted by the previous group key. If a client misses any single key update, it will not be able to interpret subsequent updates.

In a stateless system, a client does not need to have received previous key updates and can immediately interpret any new key update message. A trivial example is a system in which all clients share a symmetric master key. That key never changes and is used to encrypt and decrypt group key update messages.

A stateful system requires a reliable key distribution mechanism so that clients receive all key updates. It is therefore best suited to applications which are are able to maintain semi-permanent network connections, for example. Stateless systems are useful for applications in which no such connection is available and clients will not be able to receive all update messages. Distribution of encrypted content on DVDs is an example of a system where statelessness might be required.

3.3.2 Logical Key Hierarchy (LKH)

In order to reduce load on the server and the network infrastructure, the users can be equally divided into two subgroups. Each subgroup contains half the users and

is associated with an intermediate key-encryption-key known to its members. This intermediate key can be used to encrypt the group key, and is itself encrypted with each of the users' private encryption keys. When a group rekey operation takes place, only half of the users need to receive a new intermediate key — those belonging to the subgroup to which the new member is added (or from which an existing member is removed, respectively). Those users also receive a new group key encrypted with their new intermediate key. The other half of the users only need to receive the new group key encrypted with their existing intermediate key. Thus the required bandwidth and number of encryption operations is nearly halved.

This approach can be extended by halving each subgroup again and adding a new layer of intermediate keys. In that case approximately only a quarter of the original bandwidth and encryption operations is required.

If this process is repeated at each layer until each subgroup has only two users, the number of keys involved in a rekey operation is equal to the number of layers of intermediate keys between the affected user and the group key. This tree structure is known as *Logical Key Hierarchy* (LKH) [43]. If the tree is full and balanced, the complexity of a rekey operation is $O(logN)$.

In fact it is not necessary that the tree tree structure be binary as described above. It can have any degree. Wong et al. [45] experimentally determined that the optimal degree is 4 with respect to number of key update messages and processing time.

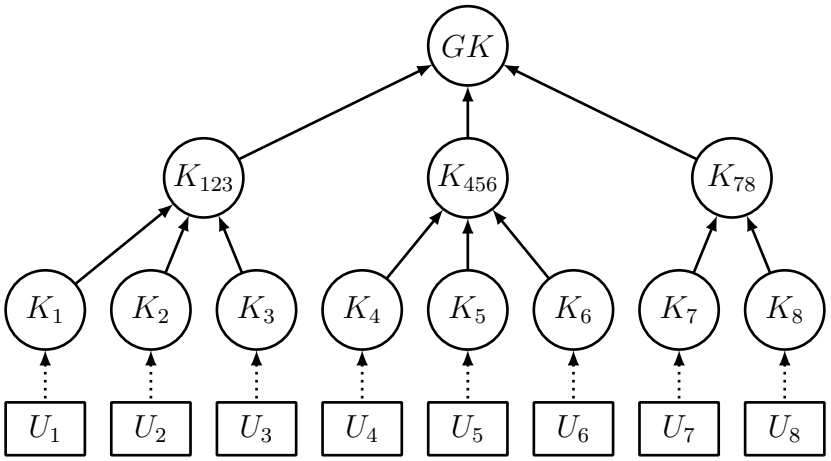
Assume that Figure 3.3a represents some initial state of an LKH tree containing 8 members, $U_1 \ldots U_8$. Each member U_i holds a key K_i known only to that member and the key server. GK is the group key shared by all members. K_{123} is known only to the members of the subtree containing users U_1, U_2 and U_3. Likewise, the key K_{456} is known to U_4, U_5 and U_6, and K_{78} to U_7 and U_8.

Given this tree structure, the following procedures are followed to update keys when a new member joins or an existing member has its access revoked.

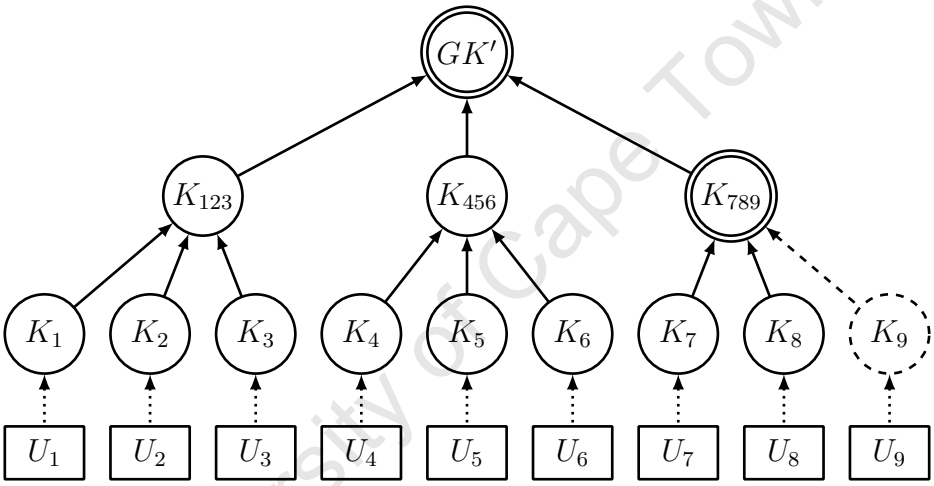
LKH Join Operation

To add a new member, U_9, the key server locates[2] an intermediate node in the tree that is not full and attaches a new child node there containing K_9. In Figure 3.3a K_{78} is this *joining point*. The key at that node and the group key GK need to be

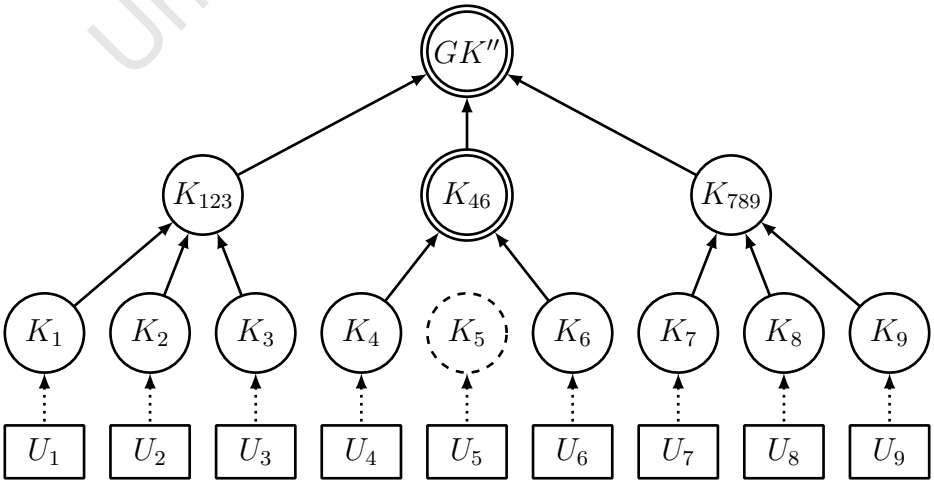
[2]Methods for locating this node are not specified by the literature and are left to a particular implementation to determine.



(a) Initial



(b) Join



(c) Leave

Figure 3.3: Logical Key Hierarchy

updated. The key server generates new random keys K_{789} and GK' to replace them, respectively. These are shown marked with double circles in Figure 3.3b. These new keys then need to be securely delivered to U_9 and to the existing members that need them. Note that initially U_9 only holds its secret key K_9.

Wong et al. [45] distinguish between three mechanisms for distributing these keys to users. These differ in how to determine the grouping of new keys and which existing keys to use to encrypt the update messages. The selection of existing keys used to encrypt new keys needs to be done in such a way as to avoid revealing new keys to users who should not hold them.

With user-oriented rekeying, the server creates a message containing exactly the keys needed by a particular user and encrypts it with a key held by that user (and other users with the same requirements). Figure 3.4 shows the key messages that would be sent to update the group key for the example given above of U_9 joining.

$$\begin{aligned} s \rightarrow U_9 &: \langle K_{789}, GK' \rangle_{K_9} \\ s \rightarrow U_7, U_8 &: \langle K_{789}, GK' \rangle_{K_{78}} \\ s \rightarrow U_1, \cdots, U_6 &: \langle GK' \rangle_{GK} \end{aligned}$$

Figure 3.4: User-oriented rekeying strategy when member U_9 joins

With key-oriented rekeying, each new key is encrypted separately, perhaps multiple times with different existing keys. Figure 3.5 shows the key messages that would be sent to perform the same group rekeying as above with the key-oriented strategy.

$$\begin{aligned} s \rightarrow U_9 &: \langle GK' \rangle_{K_9} \\ s \rightarrow U_9 &: \langle K_{789} \rangle_{K_9} \\ s \rightarrow U_7, U_8 &: \langle K_{789} \rangle_{K_{78}} \\ s \rightarrow U_1, \cdots, U_8 &: \langle GK' \rangle_{GK} \end{aligned}$$

Figure 3.5: Key-oriented rekeying strategy when member U_9 joins

In practice, implementing user or key oriented rekeying involves establishing multiple unicast connections or having multiple multicast subgroups. Group-oriented rekeying is the most straightforward to implement and results in the fewest rekey messages.

Group-oriented rekeying creates a single message containing all new keys encrypted as necessary. Members U_7 and U_8 need K_{789}. All the users need GK'. Since U_7 and U_8 already hold the previous key K_{78} and U_9 does not, K_{78} can be used to encrypt K_{789}. Likewise all existing members hold GK which can be used to encrypt its replacement GK'. These update messages are shown in Figure 3.6.

$$\begin{aligned} s \rightarrow U_1, U_2, \cdots, U_8 &: \langle GK' \rangle_{GK}, \langle K_{789} \rangle_{K_{78}} && \text{(multicast)} \\ s \rightarrow U_9 &: \langle K_{789}, GK' \rangle_{K_9} && \text{(unicast)} \end{aligned}$$

Figure 3.6: Group-oriented rekeying strategy when member U_9 joins

LKH Leave Operation

In Figure 3.3c, member U_5 is removed. Key K_{456} needs to be replaced with a new key K_{46} and GK' is replaced with GK''. However since U_5 holds both of these existing keys (K_{456} and GK'), they cannot be used to encrypt their replacements. GK'' is therefore encrypted separately with K_{123} and K_{789}. K_{46} is encrypted with K_4 and K_6. Finally, GK'' is encrypted with the new K_{46} which members U_4 and U_6 will obtain after decrypting the previous message. These update messages are shown in Figure 3.7.

$$s \rightarrow U_1, U_2, \cdots, U_9 : \langle GK'' \rangle_{K_{123}}, \langle GK'' \rangle_{K_{789}}, \langle K_{46} \rangle_{K_4}, \langle K_{46} \rangle_{K_6}, \langle GK'' \rangle_{K_{46}}$$

Figure 3.7: Group-oriented rekeying strategy for member U_5 leaving

LKH+

Perlman [32] suggests an improvement to LKH which he calls "LKH+". In the case of a join operation, existing members apply a one-way hash function to the updated keys. This means that it is not necessary to transmit the actual key data when performing such an update. Only a message indicating the key has changed is required, thus saving significantly on bandwidth.

3.3.3 One-Way Function Trees (OFT)

McGrew and Sherman [26] propose a technique based on one-way functions applied bottom-up in a key tree. The key server maintains a binary tree. Unlike LKH, interior nodes of the tree are computed based on their children. Associated with each node i of the tree are two keys: the node key K_i and the *blinded key* Kb_i. $Kb_i = f(K_i)$ where $f(x)$ is a one-way function. Kb_i is blinded in the sense that it cannot be used to determine K_i due to the one-wayness of f (without incurring great computational expense). The use of blinded keys allows a particular member to calculate the intermediate keys on its path to the root without being able to determine the node keys of its siblings.

Each leaf node is associated with a group member and the node key K_i of a leaf node is known only to that member and the key server. The node key K_j of an interior node is derived from the blinded keys of its left and right children: $K_j = f(K_L) \oplus f(K_R)$, where $\oplus$ represents a mixing function such as bitwise exclusive-or (XOR).

Each member knows only its own node key and the blinded keys of the sibling nodes on the path from its leaf node to the root. It can use these to derive the node keys on its path to the root. The node key of the root node is the group key. See Figure 3.8 for an example of how keys are derived within the tree. OFT requires less bandwidth for key update messages than an equivalent binary LKH tree. This is because updating any key only requires distribution of its blinded key to the half of the members rooted at either the key's left or right child node.

OFT Join Operation

When a new member joins the group, an existing leaf node is split into two. The existing member is placed as the left child and the new member becomes the right child. For example in Figure 3.9a member U_5 has joined the group originally shown in Figure 3.8. Some existing leaf node is chosen, in this case the one that was associated with U_3. U_3 now becomes the left child and receives a new key K'_3. Likewise the right child for the new member becomes a new key, K_5. Then the original K_3 is replaced with $K_{35} = f(K'_3) \oplus f(K_5)$, K_{34} with $K_{345} = f(K_{35}) \oplus f(K_4)$, and K_{1-4} with $K_{1-5} = f(K_{12}) \oplus f(K_{345})$.

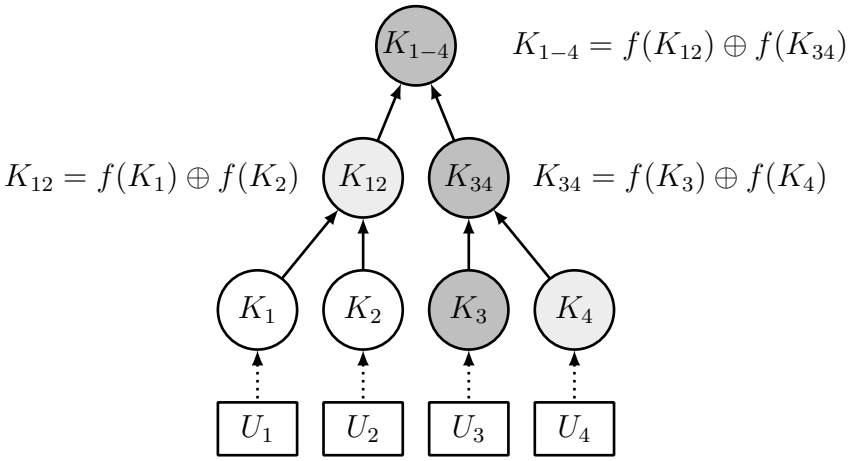


Figure 3.8: One-Way Function Tree: In this example, member U_3 holds node key K_3 and blinded keys $Kb_4 = f(K_4)$ and $Kb_{12} = f(K_{12})$. It can thus derive $K_{34} = f(K_3) \oplus Kb_4$ and $K_{1-4} = f(K_{34}) \oplus Kb_{12}$. K_{1-4} is the root key.

In general, if the key K_x for node x is updated, its blinded key needs to be distributed to the members in the subtree rooted at its sibling node s. These members already hold the node key K_s which can be used for encryption.

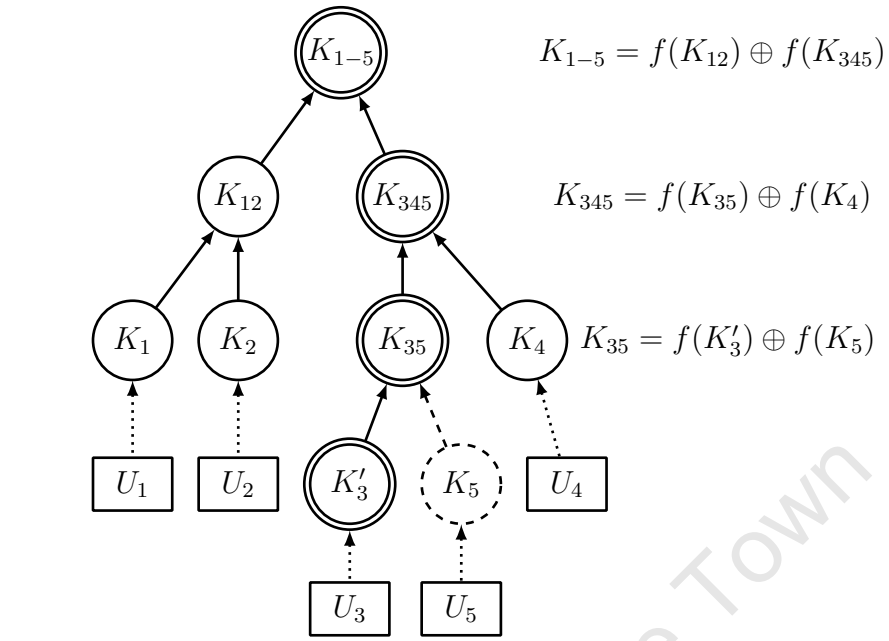
Therefore $Kb_{35} = f(K_{35})$ will be encrypted with K_4 and $Kb_{345} = f(K_{345})$ with K_{12}.

K'_3 can be encrypted with K_3 as only the security of its blinded key is compromised for further use. It is assumed that K_5 and other blinded keys required by U_5 are transmitted as part of some separate secure communication. The updated keys would be distributed as follows (K_{s5} is some arbitrary key known only to the server and member U_5):

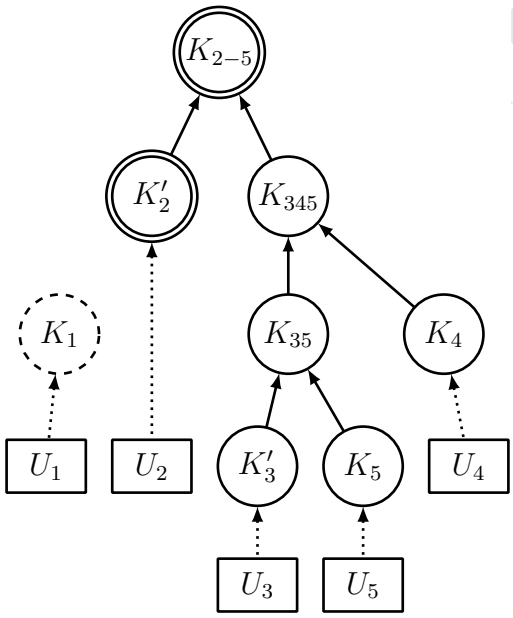
$$s \rightarrow U_5 : \langle K_5, Kb'_3, Kb_4, Kb_{12} \rangle_{K_{s5}} \quad \text{(unicast)}$$
$$s \rightarrow U_1, \cdots, U_4 : \langle K'_3 \rangle_{K_3}, \langle Kb_{35} \rangle_{K_4}, \langle Kb_{345} \rangle_{K_{12}}$$

OFT Leave Operation

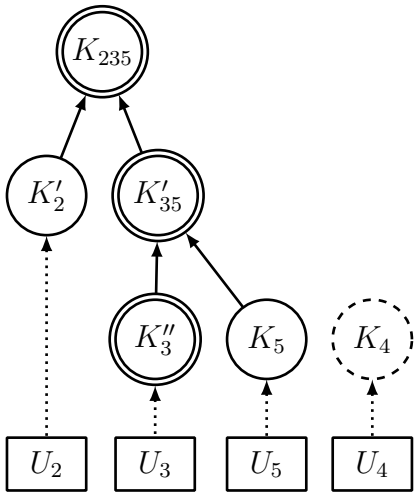
When a member leaves, the update process is similar. It is necessary to differentiate between two cases. In the first case, the sibling of the leaf node associated with the removed node is also a leaf node. In this case, the member associated with the sibling is reassigned to the parent node and given a new key. For example, Figure 3.9b shows U_1 being removed from the group shown in Figure 3.9a. The nodes K_1 and K_2 are removed, and U_2 is reassigned to its original parent node, and given a



(a) Join



(b) U_1 leaves



(c) U_4 leaves

Figure 3.9: OFT: Member join and leave operations. In (a) U_5 joins. Then U_1 is removed in (b) and finally U_4 removed in (c). Nodes marked with double circles have changed. Their respective blinded keys need to be distributed to the subtrees rooted at their siblings.

new key K_2'. The key update message is multicast as follows:

$$s \rightarrow U_2, \cdots, U_5 : \langle K_2' \rangle_{K_2}, \langle Kb_2' \rangle_{K_{345}}$$

In the second case, the sibling is an intermediate node — the root of a subtree. The parent node is then replaced by the sibling and one of the leaf nodes in the subtree needs to be changed so the node key (and therefore the blinded key that was known to the removed member) is updated. Figure 3.9c illustrates this by removing U_4 from the group. K_3' is the leaf node in the sibling subtree that is arbitrarily selected to be changed (in order to change the blinded key of K_{35} that was know to U_4). The update message would be as follows:

$$s \rightarrow U_2, U_3, U_5 : \langle K_3'' \rangle_{K_3'}, \langle Kb_3'' \rangle_{K_5}, \langle Kb_{35}' \rangle_{K_2'}$$

3.3.4 Subset-Difference Rekeying (SDR)

Subset-Difference Rekeying is a stateless protocol for group rekeying proposed by Naor et al. [29]. A stateless protocol is attractive because it allows for members to obtain the current group key even after having missed previous key updates due to being offline for a period of time. It works on the basis of users possessing information about other users in such a way that when a user is removed, the remaining users can recalculate the group key without the removed user being able to.

The key server maintains a binary tree, and as with the other techniques, the leaf nodes represent subscribers. Consider the two subtrees rooted at the internal nodes, or *vertices*, V_x and V_y, where V_y is a descendant node of V_x. S_x and S_y are sets of users (leaf nodes) in each of these subtrees respectively and $S_y \subset S_x$. Then S_{xy} is the subset of users contained in S_x but not S_y, that is $S_{xy} = S_x \setminus S_y$. A key is associated with each subset such that if a subset describes a selection of members, less the revoked members, it can be used to encrypt the new group key for distribution to valid members. This is illustrated in Figure 3.10.

The challenge is to partition the valid members into a minimal number of subsets and encrypt the group key with each associated key. Clearly a member will be included in many possible subsets and it is necessary to determine a way to minimize the number of keys that the member must store.

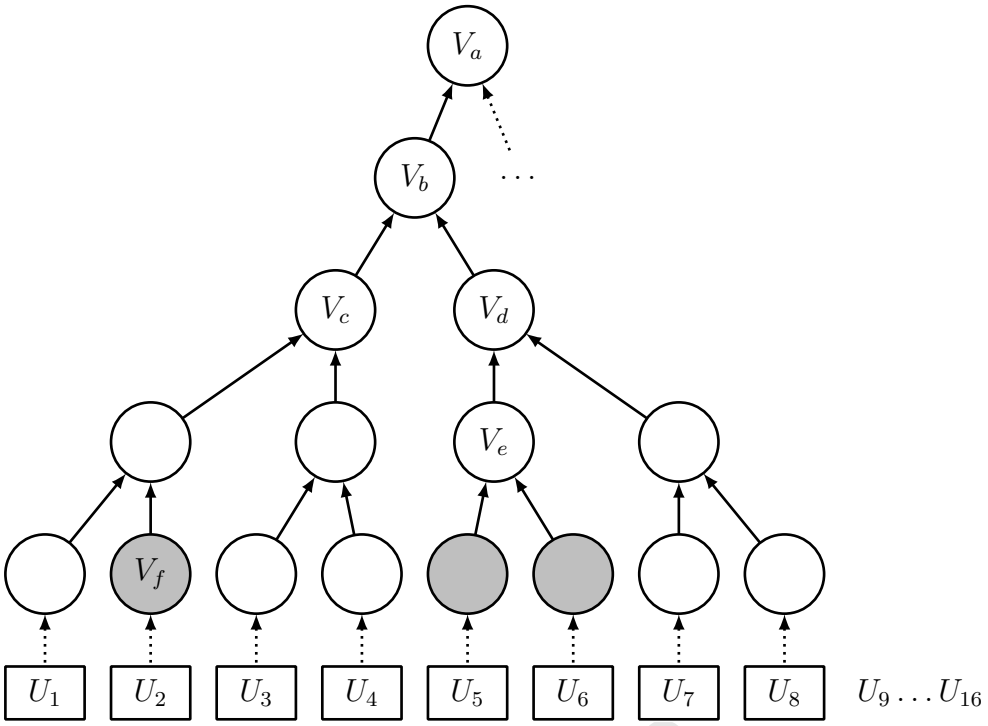


Figure 3.10: Subset-Difference Rekeying. Part of a tree containing 16 group members. Assume members U_2, U_5 and U_6 have been revoked (shown as filled nodes). The remaining members can be represented by the subsets $S_{ab} = \{U_9, \cdots, U_{16}\}$, $S_{de} = \{U_7, U_8\}$ and $S_{cf} = \{U_1, U_3, U_4\}$.

This method hinges on it perpetually retaining information about removed members. That is, the nodes associated with revoked users must remain in the tree. Thus over time as the number of revoked users grows, so will the number of subsets, and the number of separate encryptions of the group key that must take place. The server memory usage also increases with time. For a pay-TV system, it must be assumed that members will be removed frequently as they change or cancel subscriptions. Therefore the performance of the SDR technique can be expected to degrade over time.

3.3.5 MARKS

Multicast Key Management using Arbitrarily Revealed Key Sequences (MARKS) [6] takes a different approach. A binary tree is used, but the nodes are not associated with group members. Instead membership duration is divided into fixed time periods and a group key is associated with each period. The key server maintains a binary hash tree where the key associated with each node is derived from the key in its parent node by application of a one-way hash function. Two distinct functions f and

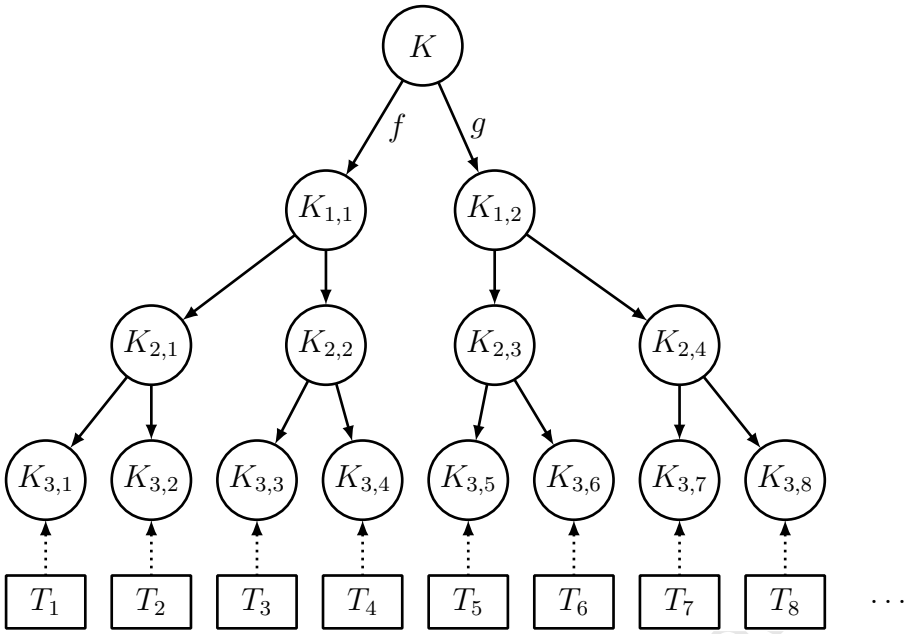


Figure 3.11: MARKS: The children nodes at each level are derived from their parent node by applying one-way functions f and g.

g are used to derive the left and right child keys respectively. Leaf-nodes represent the group key to be used for some fixed time period. Once the time period has passed, the next successive leaf node's key is used. Members have knowledge of f and g and by providing a member with the keys of intermediate nodes he is able to derive the larger set of keys associated with his time period.

An example is shown in Figure 3.11. For a pay-per-view application we might assume the time periods shown, T_i, represent hours. A particular subscriber might request time periods T_3, T_4, and T_6. The server can then supply him with keys $K_{2,2}$ and $K_{3,6}$. He can derive group keys $K_{3,3}$ and $K_{3,4}$ from $K_{2,2}$ by applying f and g respectively. $K_{3,6}$ is already a leaf node key. In this way, the user is provided with exactly the keys required for his allocated subscription period. The number of keys actually transmitted is smaller than the number of keys actually required by the user.

3.3.6 F-PPC Key Refreshment Scheme

Sun et al. [41] propose the Flexible Pay-per-Channel (F-PPC) model which incorporates both a group rekey operation and a method for more efficiently handling the problem of multiple services ("channels"). The latter is described further in Section 3.4.2.

They claim superior performance for their binary-tree based group rekey operation over other methods. Their method relies on each subscriber *not* possessing some secret associated with itself that *is* possessed by every other subscriber. While similar to SDR, this method does not have the benefit of statelessness. To varying extents all the other rekeying methods discussed in section 3.3 depend on subscribers having access to only a very limited subset of information that is possessed by others (in the form of intermediate tree nodes).

However in F-PPC the secret associated with a subscriber has permanence. Once created, there is no method to update it for the duration of the subscription. Also, knowledge of the secret is held by all other subscribers. This makes the system particularly vulnerable to collusion between any two or more subscribers in a way that would be difficult to stop without regenerating the entire tree. Our presentation of an example attack follows.

Basic collusion attack on F-PPC rekey operation

Say an attacker purchases two independent subscriptions, A and B for the purposes of this example, with full access to all services. If the attacker can somehow retrieve from B the secret associated with A, he can then unsubscribe both A and B and still be able to obtain the new group key that is generated after removal of A from the system, and can continue to do so in perpetuity as he now has complete information about the system. Even if this is discovered by the operator, there is no way to stop the unauthorised access to content without regenerating the entire F-PPC tree. Even so, it would not be burdensome for the attacker to repeat the process.

The security of this technique therefore hinges entirely on the security of subscriber local storage (ie. the smartcard or other encrypted storage). Furthermore, the technique is stateful and therefore provides minimal benefit over LKH and OFT. The single point of failure and the scale of potential compromise makes this method unsuitable for our purposes.

3.3.7 Comparison of Rekey Techniques

Table 3.1 provides a summary of the aforementioned group rekey mechanisms. The benefits and drawbacks associated with each technique are compared with consideration for the requirements of a conditional access system.

Technique	Benefits	Drawbacks
LKH: Logical Key Hierarchy	Efficient $O(log_d n)$ performance for key updates with n users and a tree of degree d.	Stateful. If users miss updates due to packet loss or being offline for a period of time, they need a fallback mechanism to receive keys.
LKH+: Logical Key Hierarchy+	Reduces bandwidth for key updates on a join operation over LKH.	Improvement does not apply to member leave operations.
OFT: One-Way Function Trees	No need to transmit actual key data (although blinded keys must be distributed).	Stateful (as above). Restricted to binary trees.
SDR: Subset-Difference Rekeying	Stateless technique.	Performance degrades as the size of the set of revoked users increases. Higher key storage requirements for members.
MARKS: Multicast Key Management using Arbitrarily Revealed Key Sequences	Stateless technique that allows providing users with keys for a predefined time period upfront.	No ability to revoke users who have already received keys.
F-PPC: Flexible Pay-Per Channel	Provides an efficient means for supporting multiple services.	Key refreshment scheme is vulnerable to collusion between users.

Table 3.1: Comparison of group rekey techniques

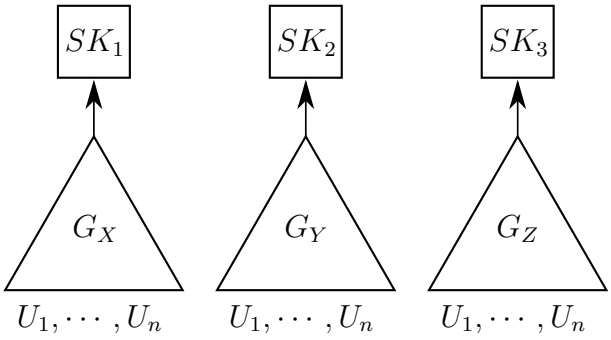


Figure 3.12: Naïve approach to supporting multiple services. A separate group key SK_i is used for each service. In the worst case, each rekey tree G_j contains all n subscribers.

3.4 Multiple Service Architecture

Thus far we have only considered the case of a single group key being shared amongst the subscribers. Typically, pay-TV operators provide multiple content services and offer a selection of subscription choices to their customers.

Whilst a single key could be used to encrypt all services, this would only differentiate between subscribers and non-subscribers. Thus any valid subscriber would technically be able to decrypt the content streams for every service. Any finer grained subscription choices would have to be enforced by the set-top-box/client software (for example, by suppressing channels which are not in the subscriber's entitlements).

Clearly this is not a cryptographically-secure approach, and is essentially "security-by-obscurity". The question arises of how to implement finer grained access controls that strictly restrict a subscriber to his/her subscription choice.

The technical limitations of any such technique should be carefully considered so as not to impose undue restrictions on the business model and service offerings of the operator.

3.4.1 Naïve Approach

The most obvious solution to the problem of securely supporting multiple services is to use a separate key tree for each service. Thus each service has a separate group key. An example with 3 services is shown in Figure 3.12.

This naive approach will be inefficient both in terms of memory and bandwidth usage. In the worst case, a system with n subscribers and m services would require memory on the order of $O(n \times m)$. Furthermore, any join and leave of a subscriber would require a rekey operation for every tree in the system, thus resulting in a potentially large number of rekey messages needing to be transmitted to subscribers.

In the case of subscription choices being divided into service groupings (known as "bouquets"), a single group key could be used for all services in that grouping. This would reduce the total number of key trees and also the number of trees that any arbitrary subscriber is a member of. If only a handful of groupings are offered, the efficiency of the system might be acceptable even in the case of a large number of subscribers and services. However this poses a specific restriction on the business model of an operator and the flexibility of its offerings. The scalability and complexity of this approach is the same as for using a separate key for each service.

3.4.2 Flexible Pay-per-Channel

The multiple service architecture suggested by F-PPC allows a subscriber to make a completely arbitrary selection of services. Subscribers are placed into groups along with other subscribers which the same selection of services. A subscriber is in exactly one such *subscription group*. Associated with each subscription group G_x is a Group Authorisation Key (GAK), GK_x. Associated with each service S_i is a group, the members of which have access to a Service Authorisation Key (SAK), SK_i. The subscription groups themselves are members of the service authorisation groups. Therefore a member of a particular subscription group uses his GAK to decrypt the SAK for each service his subscription group is authorised for. The group rekey mechanism used for distributing these GAKs and SAKs is separate from this hierarchy structure for associating groups.

Allowing completely arbitrary selections of services is a feature demanded by customers of pay-TV operators [42]. Aside from business model considerations, technical reasons limit the ease with which such a feature can be implemented.

With completely arbitrary selection of services, for each service a subscriber either does or not does subscribe to it. Therefore the number of possible combinations of m services, excluding the empty set, is $2^m - 1$. For example, if 3 services are available $2^3 - 1 = 7$ groupings of subscribers are possible. See Table 3.2 for an example of possible subscription selections of 3 services, $\{S_1, \cdots, S_3\}$, by 6 users,

$\{U_1, \cdots, U_6\}$. Their particular choices have resulted in 4 groupings $\{G_W, \cdots, G_Z\}$. The service trees and user group trees associated with these subscriptions are shown in Figure 3.13.

However with realistic numbers of services, the number of possible groupings rapidly becomes unrealistic for implementation purposes. For example, 100 services would allow a maximum of $2^{100} - 1$ possible groupings.

With F-PPC, the numbers of groupings is bounded by the number of subscribers, n. Practically $n \ll 2^m - 1$, and in the worst case, each subscriber has a unique service selection. Therefore the number of groupings is given by $max(2^m - 1, n)$. Nevertheless, an implementation of this mechanism without additional restrictions in place can easily become impractical.

3.5 Reliable Key Delivery

IP packet routing is inherently unreliable: the source transmits packets blindly without any knowledge of whether they arrive successfully at their destination(s). The concept of a "connection" is introduced by a protocol such as TCP, which facilitates a reliable *one-to-one* connection between two endpoints. TCP is therefore not applicable to multicast or broadcast transmissions. The simpler, "connectionless", UDP is used for these *one-to-many* transmissions. UDP provides no guarantees that a datagram will be received at its destination, and if it is received, there is no guarantee of correctness or that multiple packets are received in the order they were sent. The possibility of a multicast packet not being correctly received by all interested participants needs to be considered. If a member misses a key update, especially with a stateful rekeying algorithm, he will not be able to decrypt the transmission or further key updates.

In a DVB network with no return-path, certain data that needs to be readily available is repeatedly transmitted in a loop, sometimes know as a *carousel*. In comparison, on an IP network, a client who has missed out on some keys can fallback to a unicast request for those keys. Obviously it is necessary to minimize the number of such requests, otherwise the performance of the system can degrade to that where all keys are supplied individually to subscribers with unicast, which is a situation these mechanisms are trying to avoid.

	U_1	U_2	U_3	U_4	U_5	U_6
S_1	•	•	•			
S_2		•		•	•	•
S_3	•	•	•	•		•
	G_X	G_Y	G_X	G_W	G_Z	G_W

Table 3.2: Example subscriptions for users $U_1, \cdots, U_6$. Users are grouped into exactly one of $G_W, \cdots, G_Z$ along with other users who have matching subscriptions.

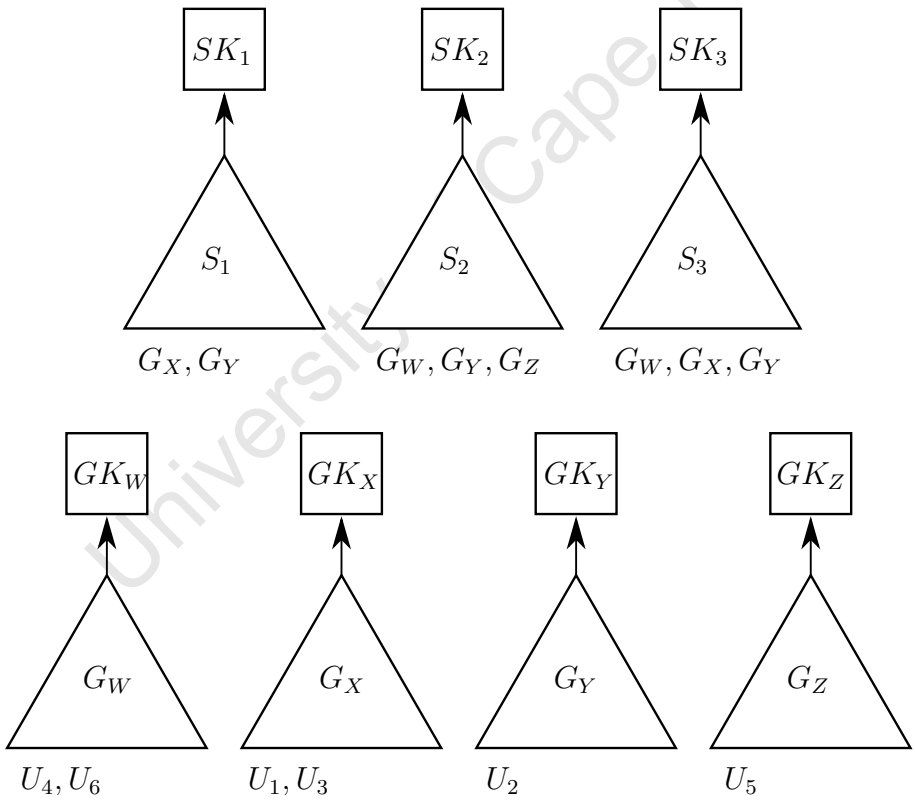


Figure 3.13: F-PPC Approach to supporting multiple services. Each user is a member of exactly one user group G_j according to his subscription preferences and receives a Group Authentication Key, GK_j. The user group is in turn a member of one or more subscription groups $S_1, \cdots, S_3$. Members of subscription group S_k will have Service Authentication Key S_k for each service their group is authorised for.

DVB Satellite, Cable, and Terrestrial networks make use of two levels of error correction [14]. Firstly, 16 bytes of optional Reed-Solomon error correction information can be added to the 188 byte MPEG transport stream (TS) packets, resulting in a packet size of 204 bytes. This provides error correction at the transport stream packet level and would remain in recorded streams, for example. Secondly, internal to the DVB transmitter a convolutional code is used with various code rates (commonly 1/2, 3/4, 5/6, 7/8). The convolutional code provides the ability to tune the error correction according to available bandwidth and network quality. This is commonly termed "FEC" (Forward Error Correction) in STB configuration menus. An FEC rate of 1/2 would mean 50% of the transmitted data represents error correction information, whereas an FEC rate of 7/8 means that only 12.5% of transmitted data is for error correction.

Various solutions have been proposed to the problem of reliable multicast networking. Examples are Pragmatic General Multicast (PGM) [39] and Scalable Reliable Multicast (SRM) [18]. These protocols either replace UDP or implement an additional layer above UDP, but they are complex [48], and often require extensive support from network equipment.

Application level solutions specifically consider the problem of reliable key distribution. These are primarily Weighted Key Assignment with Batched Key Retransmission (WKA-BKR) [38], and Proactive FEC-based Key Delivery [46] which uses Reed-Solomon erasure correction codes. Zhu et al. have proposed a hybrid method, WFEC-BKR in an extension to SDR that incorporates "self-healing" [49].

WKA-BKR improves significantly on the bandwidth usage of the *multi-send* approach wherein each key update message would be sent multiple times. A weight is assigned to each key based on the number of users that hold the key. Keys held by more users are weighted higher and are retransmitted more times than keys with lower weights.

3.6 Batched Updates

Depending on the application, it is not always necessary to immediately perform a group rekey on a member joining or leaving. This is especially the case with pay-TV. If, for example, the content is offered on a subscription basis it is only necessary to perform a group rekey operation at the end of the subscription period (say, at month end). In that case, multiple members might be revoked at the same time

and depending on the rekeying strategy being used some benefit can be gained in combining the operations. Batching key updates in this way is addressed by Setia et al. [37] and Yang et al. [46] who demonstrate reduced bandwidth usage and number of key updates.

Apart from reasons of subscription periods, and depending on the business requirements of a particular operator, the revocation of a member can be delayed until some other batch rekey frequency. The member will then be able to access content for slightly longer than their subscription period but this should not have any negative impact on the security of the system or finances of the operator if the time period between batch operations is controlled.

3.7 Secure Key Storage

A critical aspect of the security of a cryptographic system is the security of the keys themselves. The discussion thus far has considered mechanisms for securely transmitting keys over a network so that only appropriately authorised users will receive them. However once the keys are in the possession of the user, steps need to be taken to prevent their unauthorised disclosure via other channels.

The standard method of protecting keys is in a tamper-resistant device such as a smartcard. The smartcard is able to perform cryptographic operations and often the majority of client-side key handling operations are performed within it.

However smartcards incur additional costs and have limitations. Special hardware is required in a set-top-box to interface with them, and this restricts the deployment of a CA system that depends on them to devices with applicable hardware (therefore eliminating devices such as smart-phones and tablets). Royalties are often payable, and smartcards are limited in their processing power and storage capacities potentially restricting the type and frequency of cryptographic operations they can perform.

Therefore in situations where minimizing cost is a factor, or where various types of devices need to be able to receive content, a system that is secure without the need for a smartcard is necessary.

Contemporary STB hardware typically makes use of a secure boot mechanism [3]. Usually some area of the silicon in the device's system-on-a-chip (SoC) and/or its flash memory chips is able to check a cryptographic signature of the software that

will be loaded into memory. This ensures that no unauthorised (unsigned) software is able to run on the device. Together with encryption of the applicable areas of non-volatile storage (flash, or hard disk), and encryption of random access memory (RAM encryption is prevalent in STB devices), reasonable assurance can be had that an attacker will not be able to access keys stored on a STB.

This assurance is often not sufficient, however. Development of a set-top-box platform might involve multiple parties – hardware manufacturer (who also develops driver software), user interface software developers, conditional access vendor, pay-TV operator, and increasingly on modern set-top-boxes and media devices, third-party application ("app") developers. This provides multiple avenues for software that will be signed and legitimately running on the device that could potentially gain access to keys via some exploit.

Payne [31] proposes *VAULTS*, "a cryptographic access control architecture secure against privileged attackers". He describes a complete access control framework targeted at a multi-user, Unix-like, general purpose operating system. Data stored in this framework is secure even against a user with root access or physical access to the storage medium (such as hard disks). This is distinct from the usual access control scheme that uses *active protection* enforced by the kernel. Even in a system with full-disk-encryption (FDE), legitimate users might be able to elevate their privileges thus allowing unauthorised access to data belonging to other users.

VAULTS depends on the system having some form of trusted-computing or secure boot module and provides a combination of access control and a cryptographic file system. The secure bootstrap process ensures that the kernel is trusted (the process of assessing/auditing the security of the kernel prior to signing is not considered).

Briefly, each user has a secure repository known as a "vault" – an encrypted area used to store keys, fingerprints (hashes) of system programs, and "tickets" used to access cryptographically protected files. These are decrypted at login and their contents are only accessible by cryptographically verified processes. The secure kernel prevents access to vaults when they are stored in memory. Apart from user vaults, various system vaults and a public-key infrastructure are defined. The kernel acts as an intermediary, decrypting files on behalf of users who hold valid tickets. Users never have direct access to the actual protection keys.

3.8 Summary

In this chapter, we have presented a comprehensive survey of techniques that facilitate a key server securely sharing and frequently updating a Group Key amongst a large, dynamic group of receivers. In Section 3.2 we introduced the concept of a hierarchy of keys to facilitate scalability of rekeying operations. In Section 3.3 we reviewed and categorized existing techniques for group rekeying: the stateful "Logical Key Hierarchy" (LKH), LKH+, and One-way Function Trees (OFT), and the stateless Subset-Difference Rekeying (SDR) and Multicast Key Management using Arbitrarily Revealed Key Sequences (MARKS).

In Section 3.4, we considered techniques for expanding the group rekeying techniques to a pay-television system that cryptographically separates users with different subscription selections. In Section 3.5, methods for addressing missed or corrupted key updates were outlined. In Section 3.6, batched keys updates are introduced as a trade-off that sacrifices the security of immediate rekey operations for a reduction in the number of key updates that need to be performed. Finally, in Section 3.7, we discussed approaches to securing the persistent storage of keys by the receiver.

Chapter 4

Proposed CA System

4.1 Overview of Proposed System

In order to answer the research questions posed in Section 1.3, a framework for a conditional access system has been designed. An implementation of the key management and distribution scheme is used to experimentally assess its viability.

The flexibility and efficiency of this system is facilitated by the hierarchy of keys described in Sections 4.2 and 4.3 and the particular groupings of services and users that are detailed in Section 4.4.

Each of the groupings of services and users is based on the LKH+ key-tree scheme that incorporates a one-way hash function for reducing the size of join messages. Two alternative methods are presented for "growing" a full LKH tree. The preferred method that is proposed reduces the number of messages that need to be sent when growing the tree and performing key updates for joining subscribers.

The question of supporting multiple devices (multi-room viewing scenarios) on a single subscription can be addressed with an extension to the proposed key hierarchy. This is given in Section 4.4.3. A protocol for initially establishing a secret key shared between the key server and a receiver device is given in Section 4.5.

Specific details of key server operations (Section 4.6) and the format of rekey messages (Section 4.7), along with client processing of updates (Section 4.8), and a client fallback protocol for handling missed updates (Section 4.9) are also presented.

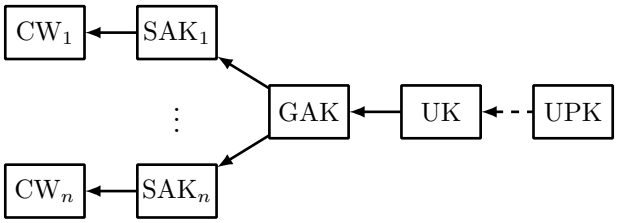


Figure 4.1: Key Hierarchy

The solution optionally incorporates batched key updates for reducing bandwidth and the total number of key processing operations. The method for achieving these batched updates is described in Section 4.10.

Suggestions are given for exploiting the properties of multicast packet routing. Greater efficiency in network utilisation can be achieved by assigning each of the aforementioned groupings of subscribers and services to separate multicast groups. Security can be improved with network support by adding access control to multicast groups. This prevents illegitimate users "piggy-backing" off the network.

Finally, to address the question of secure storage of keys without the need for a smartcard, a structure of "application vaults" is presented in Section 4.12 that relies on a trusted platform.

4.2 Key Hierarchy

A symmetric User Private Key (UPK) is shared between a subscriber and the server and known only to these two parties. The mechanism by which such a key can be shared can vary. For example, it can be a pre-shared key inserted into a set-top-box's non-volatile memory at the time of manufacture, or a public key negotiation protocol might be used to establish it (see Section 4.5 for details of such a protocol). This symmetric key would typically remain unchanged for the duration of the subscription. It is used to share a User Key (UK) associated with the leaf-node of a LKH tree that is also known only to the server and the subscriber. The UPK allows for updating the UK when necessary (eg. if a user changes his subscription selection) and for sharing additional session keys (eg. for unicast communications such as video-on-demand).

The user key is used to decrypt a Group Authentication Key (GAK), a group key shared amongst users with the same viewing preferences (ie. matching subscription

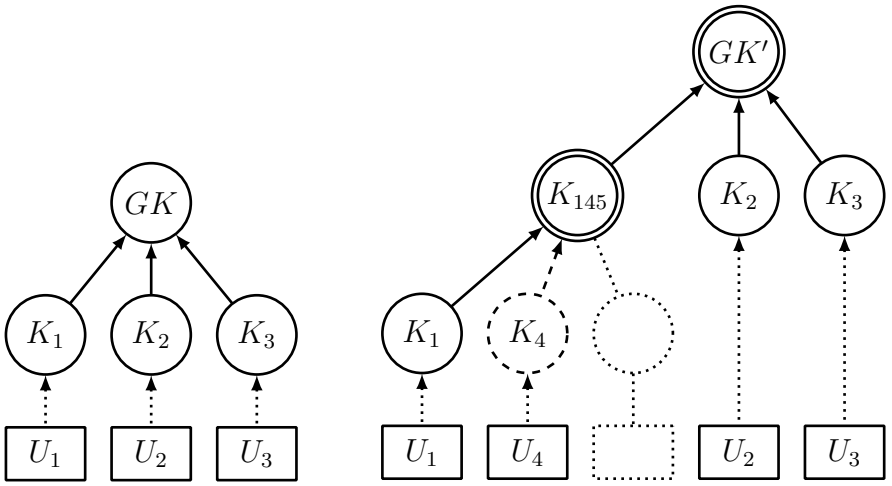
selections). A user is placed into exactly one such group. This means the total overhead of this multiple group system is minimal compared to placing all users in a single group. The GAK is then used to decrypt another group key, the Service Authentication Key (SAK). This supports multiple independently encrypted services: groups of users are authenticated for specific services.

To maintain compatibility with DVB descrambling equipment and to fit into the DVB-CA framework, the actual data stream that is encrypted by our system is the sequence of control words (CWs) used to scramble the video. These control words are updated frequently (as often as every few seconds). The SAK is used to decrypt the control words of the service associated with it. The encrypted CW is transmitted along with the encrypted video content in ECMs. It can be updated as often as the operator likes, without needing the more expensive group rekey operations to be performed so frequently. Further, the distribution of these CWs is stateless — if any particular CW is missed by a receiver, it will be able to start decrypting as soon as it receives the next CW (typically within a few seconds), thus resulting in minimal disruption.

The presented structure consisting of the User Key, Group Authentication Key, Service Authentication Key, and Control Word represents a 4-level key hierarchy. The keys associated with a particular user are illustrated in Figure 4.1. A key refreshment scheme is associated with each of these keys. The control words and user keys are shared and updated as described above. The GAKs are updated via key trees associated with each group. Likewise, the SAKs are updated via key trees associated with each service. The key tree mechanism is described in Section 4.3. The relationship between GAKs and SAKs is defined by a multiple service architecture described in Section 4.4.

4.3 Group Key Refreshment

The implementation makes use of the LKH+ algorithm, which in practice is found to perform better than OFT [6]. This is due to OFT using a binary tree (degree of 2), whereas LKH is not restricted in this way. Wong et al. [45] found that the optimal degree for a LKH tree is 4, which reduces the height of the tree significantly and therefore the number of key update operations that need to be performed for any join or leave operation.



(a) Full tree **(b)** After splitting existing node

Figure 4.2: One approach to inserting a new member, U_4, into a full LKH tree.

The literature gives little consideration to the specifics of building and maintaining the structure of a LKH tree. The $O(log_d n)$ performance of a LKH tree assumes the tree is balanced, however the algorithm does not guarantee this. Haroon [19] proposes a mechanism for balancing a binary LKH tree based on AVL rotations, however this does not extend to LKH trees of variable degree.

4.3.1 Splitting Leaf Nodes

Once all leaf nodes in a LKH tree of a particular height are full, some mechanism is required for "growing" the tree to insert further members. One approach is to split an existing leaf node by moving it down one level and inserting a new intermediate node as its parent. See Figure 4.2 for an example. The node containing K_1 is split and a new key K_{145} is placed there. K_1 is moved to the left child node, and the new user's key K_4 is inserted as the first sibling. The key update message would be transmitted as follows:

$$s \rightarrow U_4 : \langle GK', K_{145} \rangle_{K_4} \quad \text{(unicast)}$$
$$s \rightarrow U_1, \cdots, U_3 : \langle K_{145} \rangle_{K_1}, \langle GK' \rangle_{GK}$$

The number of keys to be updated by this method for inserting a new member into an already full tree by this method is $O(log_d n)$, assuming the tree is balanced initially.

4.3.2 Subtree Insertion Approach

The novelty in the method used in this implementation is to generate a new root node and insert the existing tree as a subtree rooted as the first child of the new root. All existing members thus only have to receive a single key update as follows:

$$s \rightarrow U_4 : \langle GK', K_{456} \rangle_{K_4} \quad \text{(unicast)}$$
$$s \rightarrow U_1, \cdots, U_3 : \langle GK' \rangle_{GK}$$

This approach offers $O(1)$ performance in terms of multicast key updates and keys to be processed by users and maintains a balanced tree structure. The height of the tree is maintained by inserting h intermediate nodes. The height of the new tree is thus $h + 1$ as expected and capacity is d-times larger than it was prior to the grow operation. Note that the implementation only allocates memory for the nodes as it is required. Thus in Figure 4.3 memory for only 7 key nodes is allocated, as indicated by the non-empty, circular nodes.

Nodes associated with members who have left are marked as unused. On subsequent joins, these unused nodes will be re-purposed before another grow operation takes place. For the pay-TV application, the assumption is made that over time the client base will always grow, and the tree is designed to perform efficiently under this condition.

4.4 Subscriber and Service Groupings

To support multiple services, subscribers are separated into groups using a method similar to F-PPC. The problem with that technique is discussed in Section 3.4.2. Completely flexible service selection can in the worst case degrade to the situation where each subscriber has a unique selection. The performance of a key update will be $O(n)$ in that any change in membership will result in at least n update messages being transmitted. Memory consumption on the server is of order $O(2^m)$ for m services, which is infeasible for realistic numbers of services.

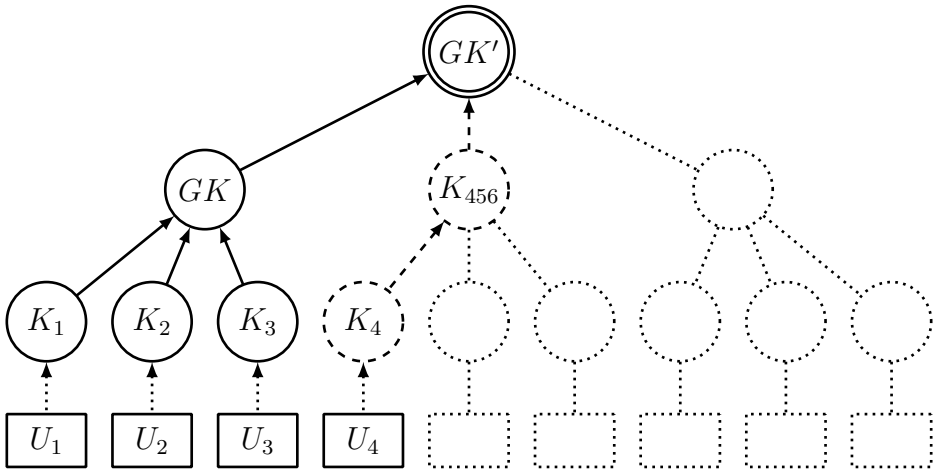


Figure 4.3: Subtree Insertion Approach. Inserting a new member, U_4, into a full LKH tree.

The solution proposed here to address these problems is to use *bouquets*. These are operator-defined groupings of services defined by DVB, usually used for convenience in package selection and STB operation. These groupings can be used to significantly reduce the number of groups of users by regarding the bouquets as atomic units of subscription that contain one or more services. Bouquets may overlap — that is, multiple bouquets might contain the same service — which prevents sharing a key between all services in the same bouquet.

Subscription groups are defined according to the possible combinations of these bouquets, which are treated as sets of services. As an example, suppose 3 bouquets are available as follows: $B_{movies} = \{S_1, S_2, S_3\}$, $B_{sport} = \{S_4, S_5, S_6, S_7\}$, and $B_{news} = \{S_8, S_9\}$. Each of the 7 possible subscription groups is associated with the union of the respective bouquets. For example, users subscribing to both "Sport" and "Movies" would be placed into the group authorised for the services contained in $B_{sport} \cup B_{movies}$.

4.4.1 Simulating Flexible Service Selection

As an aside, we provide a suggestion for simulating flexible service selection. This suggestion is compatible with the proposed multiple service architecture. An operator would start by defining a number of service groupings based on popularity. For example, the most popular services will be in more groupings while the least popular services will be in fewer groupings. These groupings could be refined over

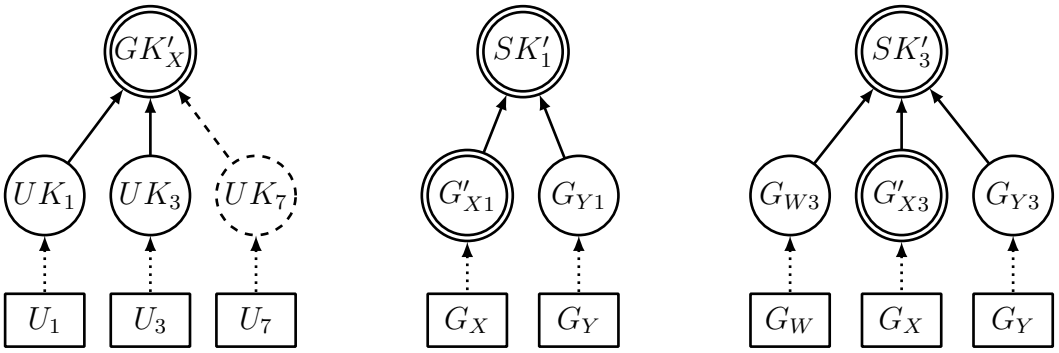


Figure 4.4: Multiple Service Architecture: Example of user U_7 joining with subscription $\{S_1, S_3\}$. Keys in nodes marked with a double circle are updated.

time based on subscription history, and will potentially be quite numerous. When a new subscriber joins with a selection of services S, find a grouping G such that $S \subseteq G$. The objective is to minimise the size of set $G \setminus S$ so that the subscriber has minimal access to services for which he is not paying. Receiver software could be used to enforce the subscription by disallowing these services.

4.4.2 Example

This description expands on the example subscriptions given in Table 3.2 and Figure 3.13. $S_1, \cdots, S_3$ are services and the SK_i keys are Service Authentication Keys. Suppose a new subscriber U_7 joins with subscription $\{S_1, S_3\}$. He is therefore allocated to group G_X. The key trees involved are shown in Figure 4.4.

The LKH+ join algorithm is followed as described in Section 3.3.2, so an updated GK_X will be multicast to users. It is also necessary to update the group keys for each of the services for which G_X is a member, thus SK_1 and SK_3 will be updated. The leaf node keys associated with G_X in each of those trees will be updated, and any intermediate keys on the path to the root.

Key updates will be delivered as follows:

$$s \rightarrow U_7 : \langle GK'_X, G'_{X1}, SK'_1, G'_{X3}, SK'_3 \rangle_{UK_7} \quad \text{(unicast)}$$
$$s \rightarrow U_1, \cdots, U_6 : \langle GK'_X \rangle_{GK_X}, \langle G'_{X1} \rangle_{G_{X1}}, \langle SK'_1 \rangle_{SK_1}, \langle G'_{X3} \rangle_{G_{X3}}, \langle SK'_3 \rangle_{SK_3}$$

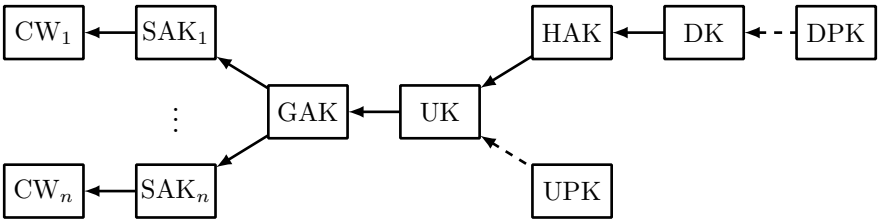


Figure 4.5: Extended Key Hierarchy to support Multi-Device Subscriptions

Note the distinction between *member* and *subscriber/user*. The system architecture uses multiple trees to support multiple services. A member of a tree may be either a subscriber or a group of subscribers.

4.4.3 Multi-Device Subscriptions

Modern households often have multiple potential viewing devices: televisions in more than one room, and mobile devices such as tablets and smart-phones, for example.

An operator may wish to enable a single subscription to access content on these various devices, possibly for an additional fee per device. This should still maintain the security properties of the system. It must be possible to revoke access for individual devices.

We propose a further extension to the F-PPC service grouping model to allow this, by introducing another level in the key hierarchy for multiple device subscriptions.

Devices associated with a particular subscription are grouped together and share a Household Authorisation Key (HAK). Each device has an associated Device Key (DK). The HAK is used to decrypt the User Key.

Additional considerations need to be implemented to ensure that devices on a single subscription are being used within the subscriber's household (or other authorised location). Various approaches to this are possible. For example, devices might need to be able to communicate with each other on a local network. Alternatively, the ISP could verify the devices are connecting from the same network endpoint.

The expansion of this idea and a possible cryptographic solution to ensuring devices are in the same household is relegated to future work.

4.5 Initial Key Establishment

It is assumed that the client and server share a symmetric encryption key, the User Private Key (UPK). The means of sharing this key are dependant on the particular application and operator's requirements. Keys might be preshared or exchanged through a secure negotiation protocol. These approaches are discussed below.

4.5.1 Preshared Keys

For purpose specific devices such as STBs, the simplest case is that a unique key is installed on the device in the factory. Alternatively, the system-on-a-chip will have a unique key associated with its serial number, both of which are known to the manufacturer and supplied to the operator.

4.5.2 Protocol for Dynamic Key Establishment

Mobile devices such as smart phones and tablets are increasingly popular and prevalent. The trend is towards supplying media to these devices. Such media would be accessible via a third-party application installed on the device.

It is necessary for the server and the application to share a symmetric key. However the server has no prior knowledge of the device. It needs some way of verifying that the application is trusted before sharing this key. We assume that a trusted kernel (or other secure component of the operating system) is running on the device. The kernel should be able to verify the integrity of applications before executing them, and acts as a trusted intermediary in the following protocol.

$$A \rightarrow S : C_A, C_K, N_A \tag{1}$$

$$S \rightarrow K : \left\langle \langle K_{AS}, N_A \rangle_{KU_A}, A, C_S \right\rangle_{KU_K}, \left\langle H\big(\langle K_{AS}, N_A \rangle_{KU_A}, A, C_S\big) \right\rangle_{KR_S} \tag{2}$$

$$K \rightarrow A : \langle K_{AS}, N_A \rangle_{KU_A} \tag{3}$$

A legend for the protocol entities is provided in Table 4.1. After S receives message 1, it verifies that KU_K (obtained from C_K) is signed by a trusted authority. After message 2 is received by K, it checks the message authentication code, then verifies that A is signed by S. If so, it passes the payload to A, from which A can obtain K_{AS} — which is then used as the UPK.

A	Application
K	Trusted Kernel
S	Key Server
K_{AS}	Key to be shared between A and S
KU_A, KU_K	Public keys of A and K respectively
KR_S	Private key of S
C_A, C_S, C_K	Certificates of A, S, and K, including respective public keys (X.509)
N_A	Nonce generated by A, used to identify this particular interaction
H	Cryptographic hash function

Table 4.1: Legend: Dynamic Key Establishment Protocol

The application needs to take necessary precautions to ensure the secure storage of keys that are provided to it. The trusted kernel should provide encryption services to applications as discussed further in Section 4.12.

If the local security of the device is circumvented (commonly referred to as "jail-breaking") so that an untrusted kernel executes, then the security of the protocol fails and an unauthorised application could be supplied with keys. A practical implementation of this protocol would need to take this possibility into account. Further platform-specific steps might need to be taken to disqualify such devices.

4.6 Key Server Operations

The subscribe, change subscription, and unsubscribe operations are assumed to be conducted out-of-band. For example a customer might call the pay-TV operator's call centre and provide personal information, payment details, and a unique device identifier such as a serial number.

Therefore at the time the subscriber is added to the tree, the necessary key updates are made by the server and multicast to the current subscribers. The new subscriber is not necessarily online at that time. When it comes online it must use the rekey request protocol to receive the necessary keys.

4.6.1 Dynamic Subscriptions

In the case of a mobile device, a subscription might be obtained by an "in-app" purchase mechanism (for example Apple iOS in-app purchases). A separate protocol can be defined for communicating the selected services along with payment verification information to the key server.

In this case, the protocol would supply the user with necessary keys immediately and then trigger the multicast updates.

4.6.2 Subscribe

The server creates a new subscriber entry and associates it with the leaf node of a particular subscription group tree. The symmetric key in the associated node is shared with the subscriber, encrypted with the UPK.

This process is shown in Algorithm 1: On line 2 the server attempts to match the subscription selection of the new user to an existing grouping of users. If no such group exists, the server creates a new key tree on line 4 and creates a new association with the subscription selection on line 5. The user is then inserted into the new grouping (line 6) and the grouping is authorised for each service in the subscription selection (lines 7 to 9).

If an existing grouping was found, the user is inserted into it (line 11) and each constituent service of the associated subscription is rekeyed from the node associated with the grouping (lines 12 to 15).

4.6.3 Change Subscription

When a subscriber's subscription selection changes, he is moved to the new subscriber group. Keys of services that were in the previous subscription and remain in the new one are not updated.

Suppose the subscriber's existing subscription is given by the set S_E, and his new subscription by S_N. The members of these sets are services.

These services can be divided into 3 groups, shown in Figure 4.6. $S_E \setminus S_N$ are the services no longer subscribed to: the subscriber needs to be removed from these. $S_N \setminus S_E$ are the new services to which the subscriber was previously not subscribed

Algorithm 1 Add New Subscriber

```
1:  procedure AddSubscriber(subscription, userId, userKey)
2:      userGroup ← MatchSubscription(subscription)
3:      if userGroup = ∅ then                          ▷ no matching user group
4:          userGroup ← NewLkhTree
5:          AddSubscription(userGroup, subscription)
6:          userGroup.Insert(userId, userKey)          ▷ userId joins userGroup
7:          for all service ∈ subscription do
8:              service.Insert(userGroup)   ▷ authorise new group for this bouquet
9:          end for
10:     else                                           ▷ found matching user group
11:         userGroup.Insert(userId, userKey)
12:         for all service ∈ subscription do
13:             leafNode ← service.FindLeafNode(userGroup)
14:             leafNode.Rekey
15:         end for
16:     end if
17: end procedure
```

and must be added. $S_E \cap S_N$ are services which the subscriber has retained. These do not need to be updated.

4.6.4 Unsubscribe

When a subscriber is removed entirely, the user group containing the subscriber needs to be rekeyed, as well as all associated service groups. Algorithm 2 shows the operation: the user node is looked up with an identifier and removed from its group key-tree on lines 2 and 3. This removal triggers a rekey operation in that tree from the node where the user's key was located. Then on lines 4 to 8 all constituent services of the associated subscription are rekeyed from the leaf nodes associated with the user grouping that contained the now removed subscriber.

4.7 Multicast Rekey Message

The format of the key update message as multicast by the key server is shown in Table 4.2. The total size of the message for a key update is 54 bytes.

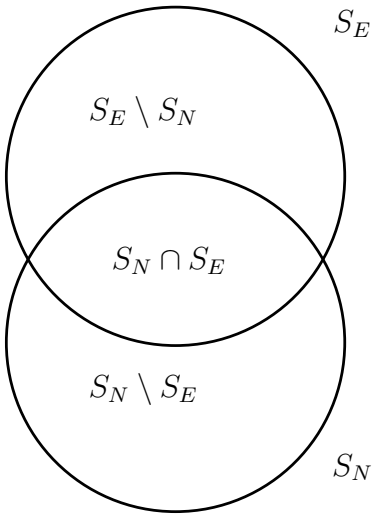


Figure 4.6: Difference in subscription sets when changing selection. Any services in the intersection of these two sets do not need to be rekeyed when a subscriber changes from existing subscription S_E to new subscription S_N.

Algorithm 2 Remove Subscriber

```
1: procedure RemoveSubscriber(userId)

2:     userGroup ← UserGroupForId(userId)
3:     userGroup.Remove(userId)

4:     subscription ←Subscription(userGroup)
5:     for all service ∈ subscription do
6:         leafNode ← FindLeafNode(userGroup)
7:         leafNode.Rekey
8:     end for
9: end procedure
```

Each key node is assigned a globally unique identifier. This identifier will not be reused across key trees. When an update message is transmitted, it includes the identifier of the key being updated, and the identifier of the key that can used to decrypt it. The identifier does not change when the key is updated. Associated with each key is a version identifier that is incremented each time the key is updated.

Size (bytes)	Name	Description
1	*type*	Update type (key data or one-way)
1	*count*	Number of individual key updates in this message
		...
4	*updateId*	ID of updated key
4	*updateVer*	Version of updated key
4	*decryptId*	ID of key used to encrypt payload
4	*decryptVer*	Version of key used to encrypt payload
16	*iv*	Initialisation Vector
16	*payload*	Encrypted Payload:
(16)	*updatedKey*	Updated Key (AES 128-bit)
		...
4	*crc*	CRC 32-bit Checksum
54	Total Size	

Table 4.2: Rekey Message Format

When using the LKH+ technique for joins, the decryption key information, initialisation vector, and payload are omitted. Therefore the size of a key update on a join is reduced to 14 bytes.

A CRC checksum is used to verify the integrity of the message. In practice this will be applied to the entire payload of a UDP packet, which might contain multiple individual key updates. In the prototype implementation a single key update is transmitted per UDP packet for simplicity.

It is assumed that for added security the multicast network infrastructure prevents clients from sending information to the multicast group. Therefore it can safely be assumed that messages are from the key server. If this is not the case, a signature needs to be added to the message for integrity verification and source authentication. Denial-of-service attacks could otherwise be made possible by the intentional distribution of bad key update messages that would corrupt the keys stored by clients.

In the scenario where the multicast network is not secured against client multicast transmissions, no circumvention of the system's security would be possible (apart from the denial-of-service already mentioned).

Each LKH+ group is assigned to a separate multicast group. Depending on subscriber distribution characteristics and network infrastructure, this can reduce network load. Primarily, however, it ensures that clients do not receive key updates related to groups they are not interested in.

4.8 Client Processing of Key Updates

A client will decrypt an update message if it holds the correct key and version that was used to encrypt it. Additionally, the client will retain update messages for keys that it already holds but for which it does not hold the appropriate decryption key. These retained messages will be reprocessed after receiving additional keys.

The logic is shown in Algorithm 3: on lines 2 and 3 the existing key and the key necessary to decrypt the update message are looked up. If the client does not hold an existing key with the same ID as that being updated in this message, but does hold the decryption key, the update message is decrypted and retained on lines 13 to 18. This represents a new key being received by the client.

Alternatively, if the client does hold an existing key with a matching ID, then the update message is processed further only if the version is newer than that of the existing key (line 5). In that case, if the correct decryption key is held (line 6) then the message is decrypted and the updated key contained therein replaces the existing one on line 8. If an appropriate decryption key is not held, the complete message is retained for processing later, after more keys have been received (line 10). This would occur if an update was missed or received out of order.

4.9 Key Recovery Fallback

The following protocol is defined for client-initiated unicast communication with the key server. This is used when the subscriber initially comes online, and subsequently in the event that the client determines it has missed some keys and needs to request retransmission.

Algorithm 3 Client Processing of Key Update Message

```
1: procedure KeyUpdate(decryptId, decryptVer, updateId, updateVer, payload)

2:     existingKey ← KeyWithId(updateId)
3:     decryptKey ← KeyWithId(decryptId)

4:     if existingKey ≠ ∅ then                          ▷ key is held by subscriber
5:         if updateVer > Version(existingKey) then
6:             if decryptKey ≠ ∅ and Version(decryptKey) = decryptVer then
7:                 updatedKey ← Decrypt(payload)
8:                 retain updatedKey
9:             else                                     ▷ keep to try again later
10:                retain decryptId, decryptVer, updateId, updateVer, payload
11:            end if
12:        end if
13:    else                                             ▷ key is not held by subscriber
14:        if decryptKey ≠ ∅ and Version(decryptKey) = decryptVer then
15:            newKey ← Decrypt(payload)
16:            retain newKey
17:        end if
18:    end if
19: end procedure
```

$$U \rightarrow S : U$$

$$S \rightarrow U : \langle \mathcal{K} \rangle_{UPK} \quad \text{where } \mathcal{K} \text{ is the set of keys required by } U$$

The client establishes a TCP connection to the server and identifies itself. The server responds by sending the updated keys, encrypted with the pre-shared symmetric key UPK. The client can discard any previously held keys and retain this new set of keys.

4.10 Optional Batched Updates

The solution incorporates the possibility to transmit key updates in batches. When a subscriber joins, any changes to keys in the tree are deferred until later (when a batch of updates is processed at once). The associated nodes in the tree are simply marked for update. The subscriber is sent the current group keys (together with

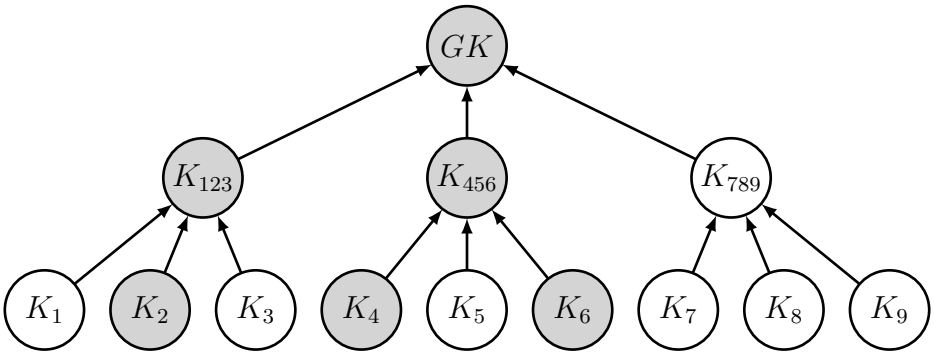


Figure 4.7: Batched Updates: Key updates are deferred at the time of join/remove. Nodes are marked for later processing. When the next batch is run they will be updated and transmitted in this order: $K_2, K_{123}, K_4, K_6, K_{456}, GK$.

the necessary intermediate keys). For removal of subscribers, the changes are also deferred. In that case, no keys need to be transmitted. When the batch is run, the tree is processed from the leaf nodes towards the root.

Figure 4.7 shows how nodes are marked for batch updates. In that example, the leaf nodes containing K_2, K_4, and K_6 are marked for rekeying. The intermediate nodes K_{123} and K_{456} are also marked along with the group key GK. When the batch is run, each of the intermediate keys and the group key is updated only once as opposed to multiple times if each rekey operation had been performed separately.

Note that batch updates impact on backward and forward secrecy. The operator can define a maximum time period t_m between batches that is suitable for their business model. Then subscribers who join or are removed will not be able to access content that was transmitted earlier than t_m before they joined, or later than t_m after they are revoked.

4.11 Multicast Group Security

Additional protections can be added at the Internet service provider's multicast network level. Specifically, an ISP possesses sufficient information to associate a particular endpoint in the network with a particular subscriber. This information could be used to restrict access to multicast groups based on the subscription associated with that endpoint.

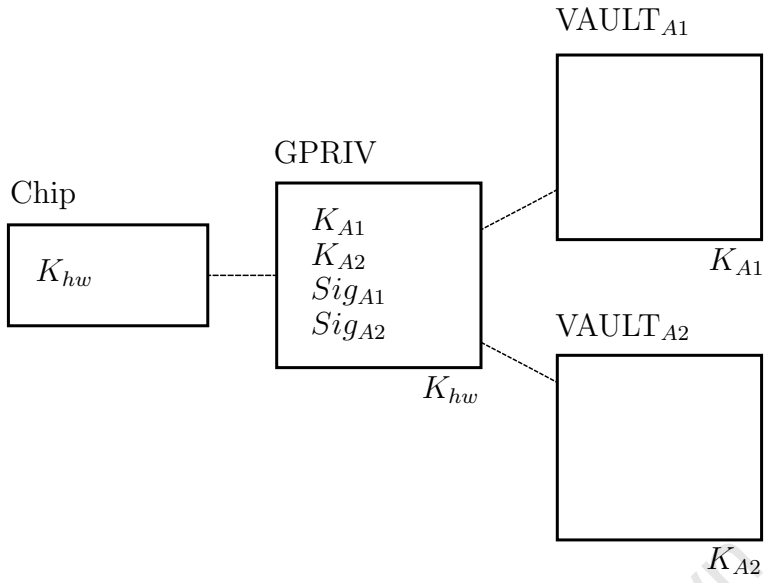


Figure 4.8: Vaults for applications $A1$ and $A2$ are encrypted by the kernel with K_{A1} and K_{A2} respectively. GPRIV, the kernel's Global Private Vault is encrypted by hardware with K_{hw}.

A restriction of this type would mean that an attacker who holds a legitimate subscription but is illegitimately redistributing content would not only have to redistribute decryption keys but also the encrypted content itself. This is because anyone who is not a legitimate subscriber would not be able to join the multicast groups used to distribute the encrypted services, even if they received the keys through some illegitimate means. This would prevent illegitimate users "piggy-backing" off the legitimate network.

4.12 Secure Storage with Application Vaults

We propose an approach inspired by VAULTS [31] (described in Section 3.7) that can be used to cryptographically separate and protect data belonging to different applications on a STB or mobile device. As in Section 4.5.2, we assume the device is running a "trusted kernel" — that is, one that is verified by hardware before being loaded.

The operating system must provide an API for applications to write to a per-application "vault". The encryption and decryption of this vault is performed on behalf of the application by the kernel. The key used for this is not available to the application.

The kernel stores a key for each application, along with cryptographic hashes of the associated application executable, in a global private vault. The kernel must verify the application's integrity by using the cryptographic hash before allowing access to the application's vault. The global private vault is encrypted with a unique key built into the device's chip. Depending on the hardware support, the kernel itself might not even be able to gain direct access to this key. A device supporting this functionality would be sufficient to allow secure storage of keys from the CA system.

Figure 4.8 depicts an example association between vaults for a system consisting of two applications, $A1$ and $A2$. Each application can store data in their respective vaults. The kernel is responsible for transparently encrypting and decrypting the contents of the vault as they are passed to and from the application, after verifying their signatures Sig_{A1} and Sig_{A2} respectively. The encryption keys K_{A1} and K_{A2} are held in the global private vault, $GPRIV$, and known only to the kernel. In turn, the $GPRIV$ vault is encrypted by hardware with K_{hw}.

4.13 Summary

In this chapter we have presented our proposed solution for the design of a Conditional Access system for IPTV. The system is designed to scale with number of subscribers and services. Each service is encrypted with an independent group key to reduce the impact of any single compromise. Further, our system allows an operator to offer different subscription packages to users and to cryptographically protect these packages in such a way that a subscriber with decryption keys for one package is unavailable to view other packages to which they are not subscribed.

Sections 4.2, 4.3 and 4.4, respectively, describe the design of our key hierarchy, refreshment scheme, and groupings of subcribers and services.

In Section 4.5, we presented a protocol for initial establishment of a key shared only between the key server and the receiving device. Sections 4.6, 4.7 and 4.8 describe the key server operations, multicast rekey message structure, and client operations that are required for our system design. A fallback protocol for recovery of missed key updates by a client is listed in Section 4.9.

In Section 4.10 our implementation of batched key updates is described. Finally, Section 4.12 describes an approach to securing the persistent storage of keys on end-user devices, particularly those that run third-party software applications.

Chapter 5

Results and Evaluation

5.1 Testing Methodology

A prototype implementation was used to conduct experiments, primarily to assess the scalability and practical feasibility of the system. The implementation is described in Section 5.2.

To generate experimental data for each test, a number of initial subscribers, N, with random subscription choices were successively added to the system. Thereafter, for bandwidth tests, the simulation randomly adds and removes subscribers in a ratio of 9:1 until the subscriber count reaches $2N$. The simulation uses 50 services divided evenly into 5 bouquets, unless otherwise specified. The subscription choice for a given subscriber is a random selection of between 1 and 5 of the bouquets. A degree of 4 was used for key trees, unless otherwise specified.

Tests were designed to assess total memory usage of the prototype server application and how it varies as the system scales. These results are shown in Sections 5.3 and 5.4.

In Section 5.5, performance of the two alternative methods for growing a full tree (as described in Section 4.3) was analysed by counting total key updates, and the *weight* of those updates. Weight was determined by counting the number of users that need to receive a particular update message.

Total bandwidth was measured outgoing from the key server and averaged for the number of add and remove operations performed in the experiment. This is shown in Section 5.6.

Section 5.7 shows the system scalability from the client's perspective. Consideration is given the total number of keys that a subscriber must hold, and the average number of key updates that the client must process when a subscriber is added or removed. Finally, the effects of batched key updates on bandwidth are shown in Section 5.8.

5.2 Implementation

A C++ prototype of the conditional access system consisting of client and server components that implement the aforementioned communications protocols and key management scheme was developed. A simulation application provides the server with multiple join and leave requests to collect experimental data.

The simulation bypasses the IP networking interface by directly triggering a large number of joins and leaves within the server module. A sample of the total clients added are simulated to monitor their key processing and storage.

For practical use, multicast client and server modules were implemented using the BSD Sockets API. A separate multicast group is associated with each group key (ie. LKH+ tree), and with each service's data stream. The data stream generated for each service consists of random or patterned data. It is expected that practical use of the system would require integrating the key server mechanism with a robust media streaming library such as Live555 [25].

All key update messages are encrypted with AES-128 in the Cipher-Block-Chaining mode of operation (AES-192 and AES-256 can optionally be used with the associated increase in key storage on the server and transmission of updated keys). A randomly generated initialisation vector is transmitted along with each encrypted key update message. For simulation purposes, key establishment is achieved by sending a unique symmetric key to each client. This is used for further cryptography operations. MD5[1] is used as a one-way function to transform a 128-bit AES key into another 128-bit key for "LKH+". All cryptographic functions are provided by the OpenSSL [30] library.

[1]MD5 was chosen somewhat arbitrarily for the purposes of the prototype implementation. The MD5 algorithm has known flaws and it is expected that a production implementation would prefer a more suitable algorithm. Such a choice should consider the the availability of hardware acceleration for these algorithms on the targeted platforms. The use of a different hashing algorithm will not affect the algorithmic complexity of the proposed group rekey operations.

Experiments were performed on an Apple MacBook Pro with a 2.66 GHz Intel Core i7 processor and 8 GB of RAM.

5.3 Server Memory Consumption

The graph in Figure 5.1 shows the memory consumption of the server application as the number of initial subscribers increases to 50,000. Memory consumption is measured using the Mach kernel's `task_info()`[2] utility function. Total resident memory of the process is sampled once per every 1,000 users that are added. The observed memory consumption was consistent across multiple runs.

The graph shows linear memory memory consumption, $O(N)$ for N subscribers, as is expected. The actual total is approximately 14.5 MB for 50,000 users (degree 4). Projecting this to 10 million subscribers, as an approximation of a large operator, memory usage would be 2.9 GB which is can easily be accommodated in a suitable server's random access memory. Inflections in the graph are visible at approximately 14,000 and 35,000 subscribers. The implementation does not alter it's behaviour at these points. The prototype implementation makes use of C++ Standard Template Library (STL) containers. In particular, these are used for managing portions of the simulation that would not be present in a real implementation. The observed inflections appear to be due to additional memory being allocated for STL vector objects when their initially allocated capacity is exhausted.

Figure 5.2 shows how total memory consumption is related to degrees of the LKH trees. It is expected that a higher degree has lower memory consumption because fewer intermediate nodes are required. The server process was run for each of the degrees from 2 to 16 and 50,000 subscribers were added. Minimum memory usage was found to be achieved with a degree of 6, which uses 25% less memory than a binary tree. Contrary to expectations, further increases to the degree see the memory usage starting to increase approximately linearly. It was observed that even tree degrees use less memory than the adjacent odd degrees. Note that the implementation does not behave differently for odd or even degrees and the number of allocated nodes. The observed increase and the difference between odd and even degrees are due to wasted memory caused by the C++ runtime's memory management alignment rules in combination with the prototype implementation's frequent and numerous dynamic memory allocations. Using a memory pool could reduce or

[2] `http://web.mit.edu/darwin/src/modules/xnu/osfmk/man/task_info.html`

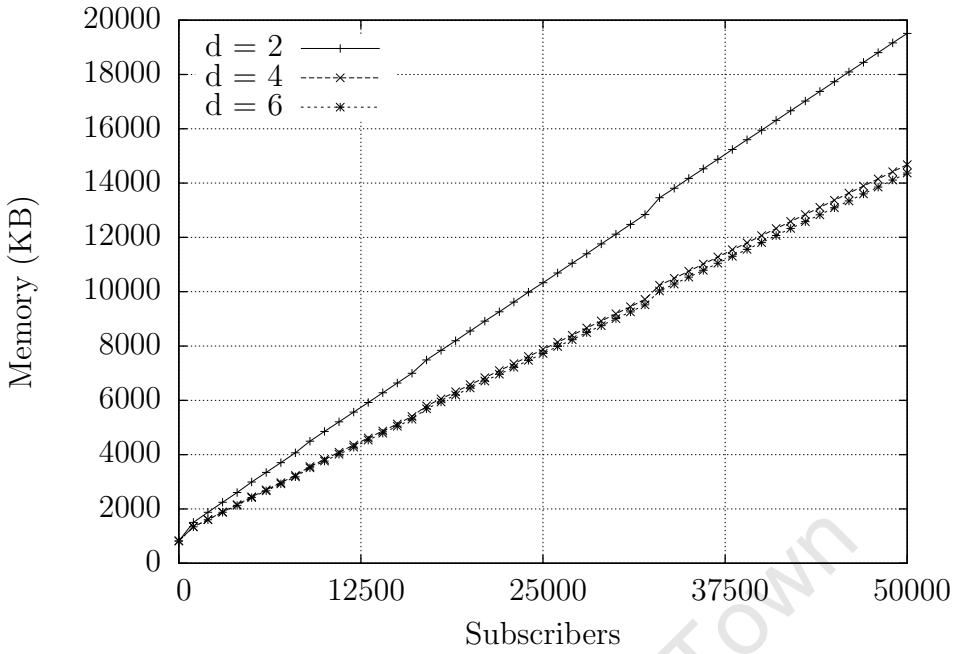


Figure 5.1: Server Memory Consumption for Varying Numbers of Subscribers

eliminate wasted space and cause the actual memory allocation to follow the total number of nodes as expected. This unexpected memory usage should therefore be considered an artifact of the prototype implementation and is not significant with respect to assessing the

The total number of nodes N in a single full tree of height h and degree d is given by the standard formula:

$$N = \frac{1 - d^{h+1}}{1 - d}$$

Where the height h for n members is given by:

$$h = \log_d n$$

Figure 5.3 compares the total number of nodes allocated for 50,000 subscribers to the number that would be needed for a single tree. The graph shows that increasing the degree causes the total number of nodes to tend towards the minimum required which would be 50,000 nodes (no intermediate nodes). The overhead of additional keys to support the multiple service architecture is shown to be approximately constant and minimal. This is because of the system property that restricts a subscriber to exactly one grouping. Dividing N subscribers amongst multiple trees uses approximately

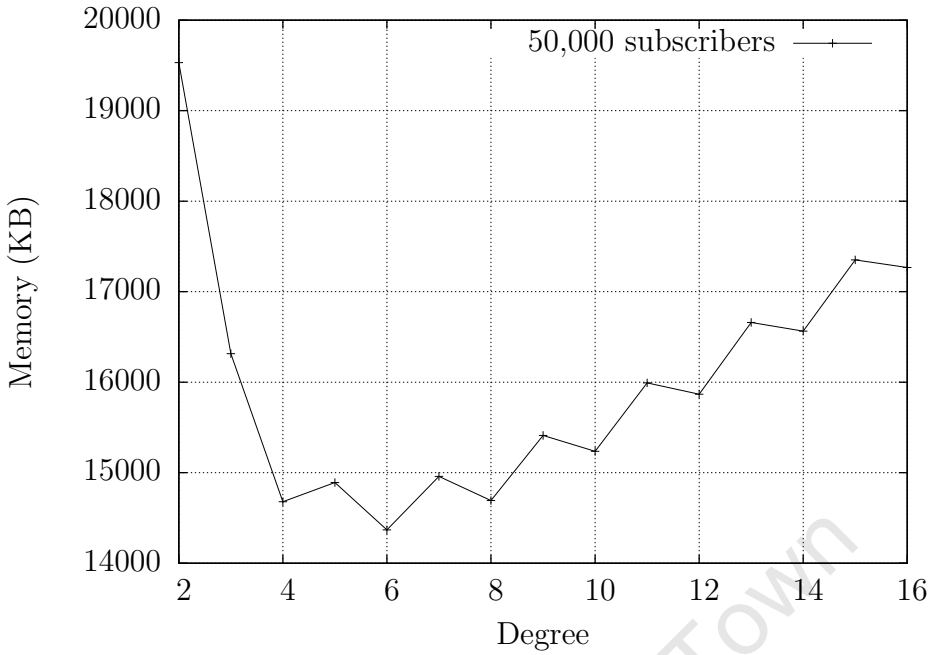


Figure 5.2: Server Memory Consumption for Varying Degrees of the LKH Trees

the same number of nodes as a single tree containing all N subscribers. The minimal overhead comes from the nodes comprising the service trees and does not present a scalability problem.

5.4 Effect of Service Groupings

Figure 5.4 shows the effect of varying numbers of bouquets. The memory usage increases exponentially: $O(2^n)$ for n bouquets. This is as expected due to the $2^n - 1$ possible combinations of bouquets that a subscriber may choose from. The linear projection on the graph demonstrating the naïve service grouping approach was calculated by multiplying the size of tree containing all subscribers by the number of bouquets for the datapoint.

Figure 5.5 shows the effect of varying numbers of services. The mean results of two runs of the simulation were used to produce this graph. The increase in memory consumption with increase in the numbers of services is negligible (for realistic numbers of services relative to the number of subscribers). The only overhead from adding an extra service is the associated key tree, which is relatively small compared to the user grouping trees. The minimal absolute memory consumption and the very

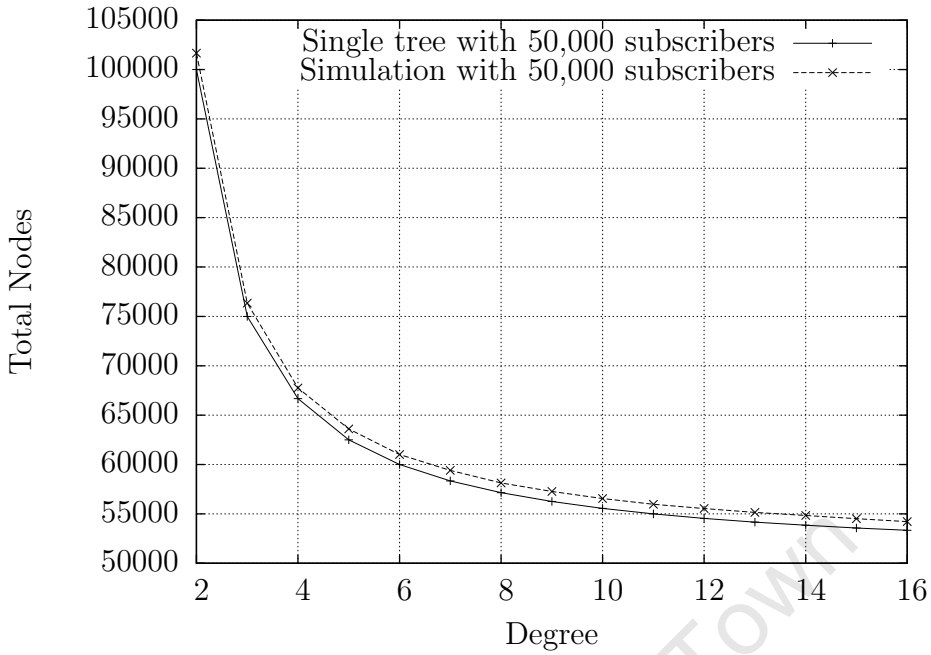


Figure 5.3: Total Nodes to support 50,000 subscribers for varying degrees of the constituent trees. Compared against the number of nodes for a single tree of 50,000 subscribers.

gradual slope of the graph that is apparent indicates that the proposed system is highly scalable with number of services. The inflection visible at 60 services is not significant: it is combination of the few runs performed and the random customer behaviour of the simulation.

Bouquets allow a large number of services to be made available in a flexible manner, since memory consumption for a given number of bouquets is linear with increase in the number of subscribers. However, it is only practical to use a small number of bouquets.

If more flexibility is required, for example a large number of bouquets, or completely flexible service selection, then using the naïve service grouping method as discussed in Section 3.4.1 will be more efficient. In that case, users would be members of multiple groups, each of which are either associated directly with a service, or with a bouquet. The implementation does not simulate this, but linear memory consumption is projected onto the graph in Figure 5.4 and shows that memory consumption of the naïve approach matches the proposed technique for 15 bouquets.

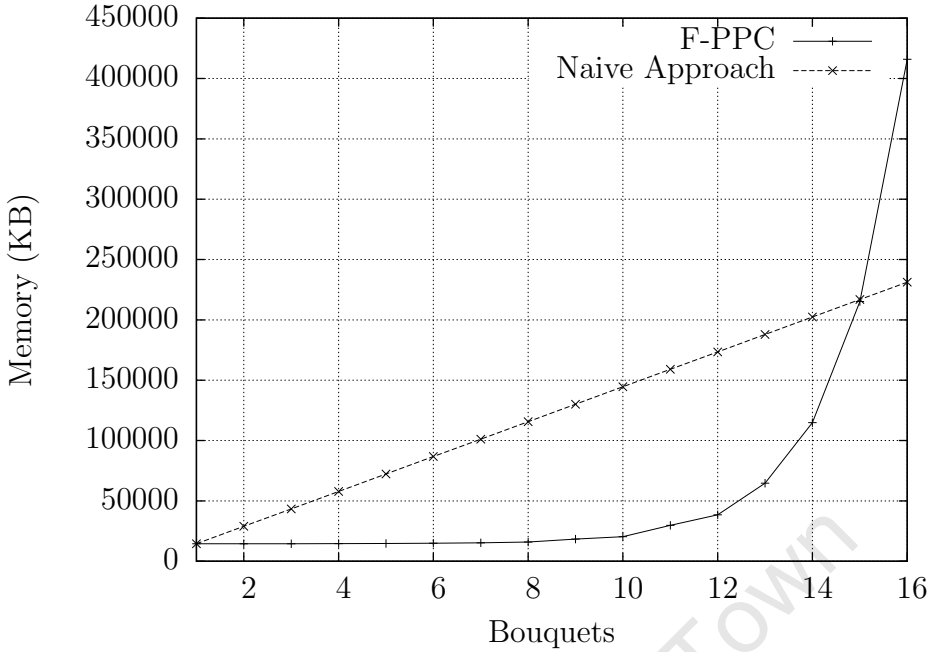


Figure 5.4: Server Memory Consumption for Varying Numbers of Bouquets. 50 services are divided evenly amongst these bouquets. For n bouquets, there are a possible $2^n - 1$ service groups. Also shown is an approximation of linear memory consumption for the "naïve approach". For $n > 15$, this becomes more space efficient than F-PPC.

It should be noted that the numbers and graphs for varying numbers of bouquets are dependent on the distribution of services amongst bouquets and the rules or trends for subscribers' selections. In the simulation, bouquets are non-overlapping and subscribers choose some or all of them. If bouquets are primarily overlapping, for example for different levels of subscription ("Compact", "Standard", "Premium"), subscribers will choose only one or a few and the exponential memory usage would not be seen.

5.5 Comparison of Tree Growth Operations

To assess differences in performance between the two approaches to growing a full tree (splitting leaf nodes and subtree insertion) 50,000 members were added to a single tree and the total number of key updates were counted. Figure 5.6 shows that subtree insertion results in fewer total key updates than splitting leaf nodes. This improvement is due to the single update message required to replace the group key when the tree grows in the subtree insertion approach.

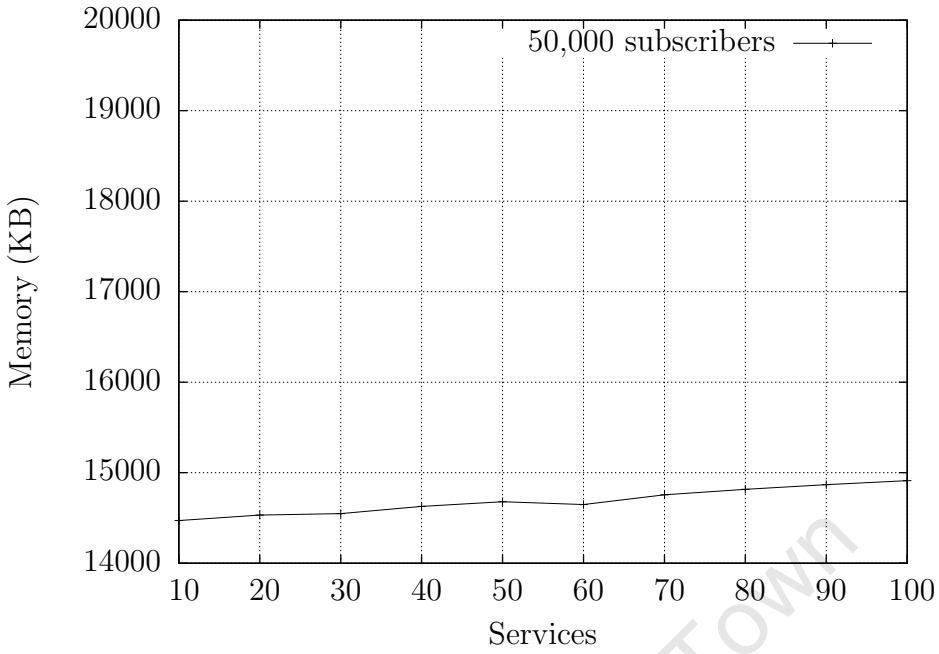


Figure 5.5: Server Memory Consumption for Varying Numbers of Services. Services are divided evening amongst 5 bouquets.

Additionally, for each update the number of users that require the key is calculated. This is the "weight" of the key update. Fewer users affected by a key update is preferable. Subtree insertion performs marginally better in this case too, as shown in Figure 5.7. This is because new subscribers are inserted into subtrees that are initially empty and so joins after a grow operation affect fewer existing subscribers. The minor inflections in this graph are again the result of simulated variability of subscriber behaviour. Apart from such minor fluctuations, results were consistent across successive runs.

While the subtree insertion operation supports rapidly growing the tree, as presented in this thesis it does not cater for shrinking the tree again in the case of a large number of members being removed. While memory associated with user keys is freed when those users are removed, the structure of intermediate nodes remains. The performance in this case will be worse than $O(log_d n)$ for a tree containing n members. In fact it will be $O(log_d m)$, where m is the maximum number of members that was previously held by the group.

With the leaf nodes splitting approach, if all siblings of a particular leaf node are removed, the intermediate parent node can be eliminated to shrink the tree (this was not implemented). The tree will become unbalanced, and rebalancing the tree is non-trivial and potentially expensive in terms of key updates. In this implementation,

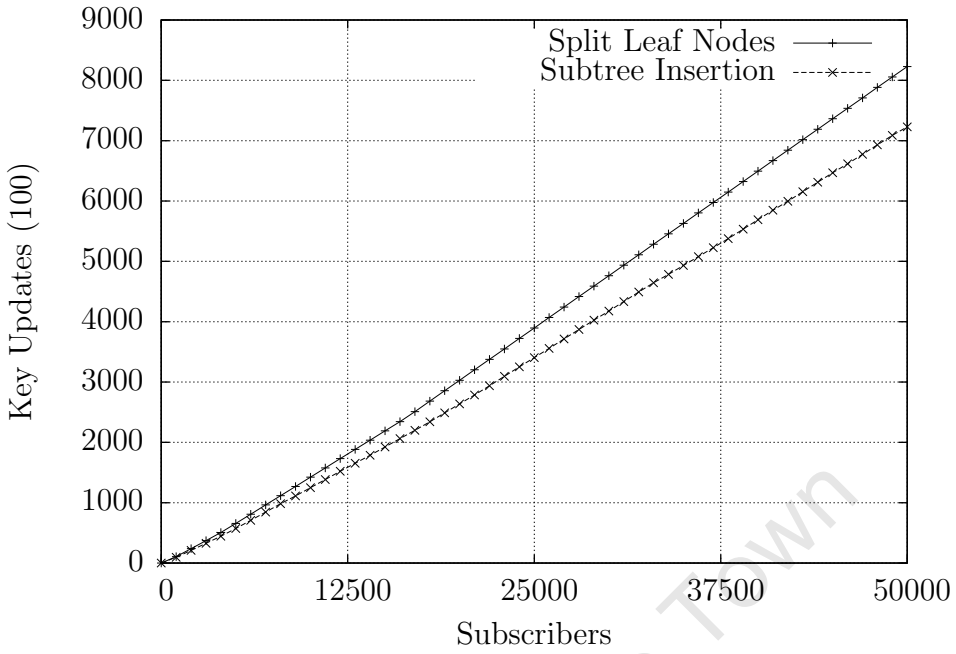


Figure 5.6: Total Key Updates: Subtree Insertion vs. Splitting Leaf Nodes. Inserting nodes into a single tree of degree 4.

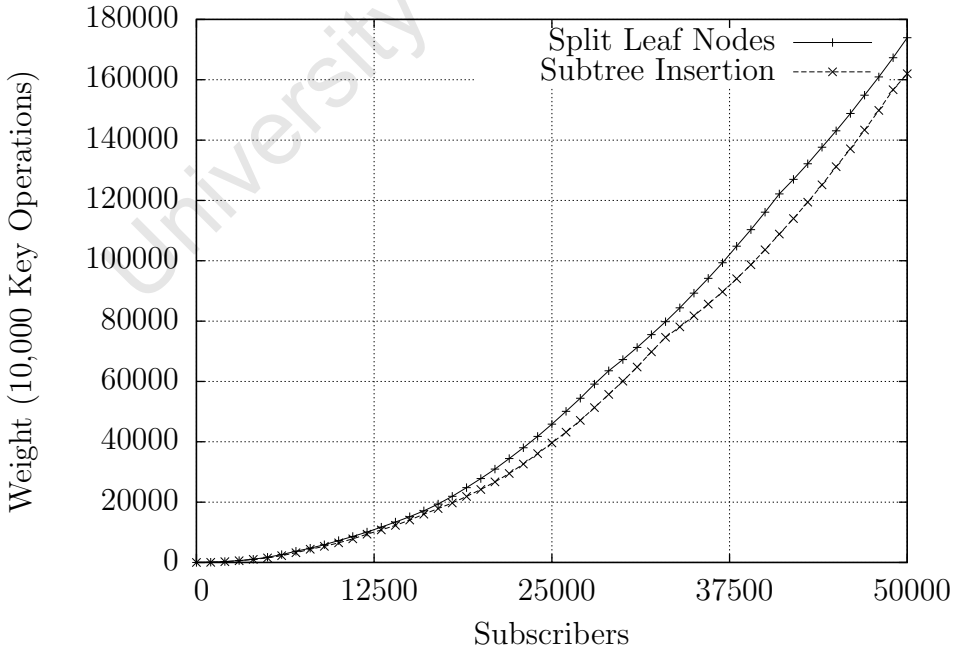


Figure 5.7: Weight of Subtree Insertion compared to Splitting Leaf Nodes. Inserting nodes into a single tree of degree 4.

	Time	
	Total (s)	Average (msec)
Subtree Insertion	3.83	0.077
Leaf Node Splitting	25.97	0.519

Table 5.1: Time to insert 50,000 subscribers.

the leaf node to split is chosen from the shortest subtree. Finding the shortest subtree causes the leaf node splitting operation to be time consuming relative to the subtree insertion approach.

5.6 Bandwidth

Bandwidth usage was calculated by increasing the number of subscribers from 50,000 to 100,000, with randomized adding and removing of subscribers in a 9:1 ratio. This simulates a pay-TV operator that is growing overall, with a 10% rate of turnover. Total data sent by the server is divided by the total number of add and remove operations. The results are shown in Table 5.2. The average data required to deliver key updates for add or remove operation using LKH trees is 13.35 KB or 4.8 KB when using LKH+ key-trees.

These figures can be compared against unicast distribution of a group key to all subscribers, or equivalently, multicasting the group key to all users encrypted with their UPK. A single add or remove operation with 100,000 subscribers would require transmitting 5,272 KB *per group key*, assuming the same update message size. This could be reduced to 1,367 KB for joins by using a one-way function. The bandwidth savings of using the proposed solution is many orders of magnitude, allowing more frequent updates to be performed.

These size calculations are based on the key update packet format defined in Section 4.7 and exclude the size of UDP and IP packet headers. This represents all outgoing key updates from the server, including initial unicast delivery to subscribers. All keys are transmitted only once, with no batched updates.

	Average Update Size (KB)			Total (MB)
	Add	Remove	Combined	
LKH	13.21	14.57	13.35	817.3
LKH+ Joins	3.71		4.81	294.7

Table 5.2: Update Size per Add/Remove Operation. The average values were calculated by dividing the total size of transmitted data by the number of add and remove operations. Total shown is the bandwidth required to increase subscribers from 50,000 to 100,000.

5.7 Client Operations

Due to the time performance of simulating a large number of clients, a 1% sample of the total number of subscribers is monitored. Fewer total subscribers were used for these experiments. The results presented are the average of two runs of the simulation which results in some inflections in the graphs due to the random simulated subscriber behaviour. The absolute values and general trends of the graphs were considered in this analysis.

The average number of keys stored by a client and the average number of updates that a client has to perform per add/remove operation are determined. Graphs are shown for varying numbers of subscribers, services, and bouquets.

Figure 5.8a shows that the average number of keys held by a user increases linearly with the number of services. The same is true for the number of key updates that a user needs to make for a single add or remove operation, on average. This is shown in Figure 5.8b. A subscriber belonging to a system with 100 services will need to hold approximately 185 keys (2.9 KB for 128-bit AES keys), which is well within the storage capabilities of any receiver device (including the storage capacity of a smartcard). The user has to perform approximately 47 symmetric decryption operations for an add/remove operation.

As the number of bouquets increase the number of subscriber groups increase, and therefore the size of the service trees increase too. A single bouquet is a special case in that all subscribers will be members of a single group, and they will also need to hold keys for all services. For other cases, a linear increase is shown in the average number of keys held by a user (see Figure 5.8c).

Apart from the special case of a single bouquet, the number of key updates performed by a user is constant with an increasing number of bouquets. This is because a user is restricted to a single service group, which is some fixed combination of bouquets.

Figures 5.8e and 5.8f show that client key operations and number of keys held are constant for increasing numbers of subscribers. In fact, the number of keys held by a subscriber within a particular tree is given by $log_4 n$, however for the ranges of subscribers being considered the differences are negligible and outweighed by the number of service tree keys that a subscriber must hold.

5.8 Effect of Batched Updates

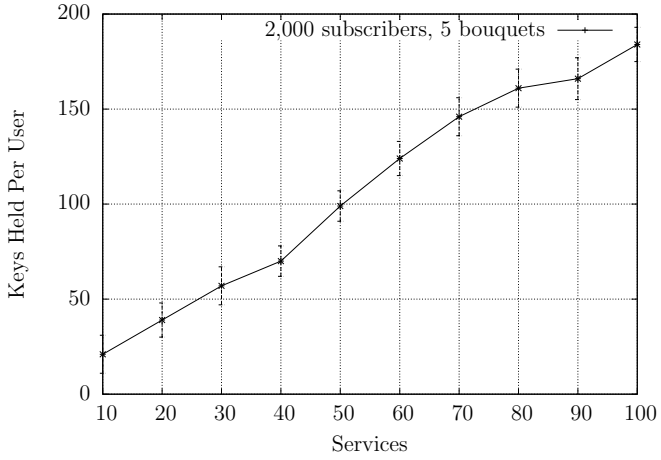
The implementation supports an optional batch mode. The effect of batched updates on outgoing server bandwidth and client operations was tested by varying the number of add/remove operations per batch. Figure 5.9 shows that bandwidth initially reduces significantly as batch size increases, then flattens out for a batch size of about 300. At that point the multicast group updates have reduced to almost zero, and the remaining bandwidth usage is for initial unicast transmission of keys to joining members.

Similarly, the average number of key updates that a client has to process drops significantly with increase in batch size as shown in Figure 5.10. Note that for any single batch update operation, the number of key updates that a client processes will not be reduced from that of a non-batch operation (and may in fact be slightly greater).

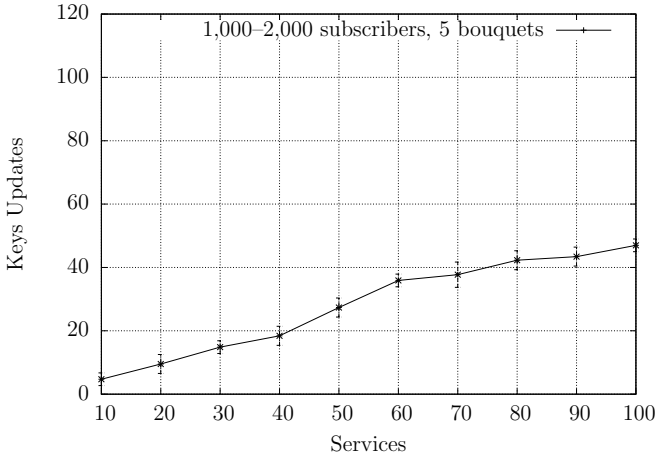
These results show that use of even moderately sized batches result in an appreciable reduction in bandwidth overhead. In a pay-TV environment, dependant on subscriber turnover rates and subscription periods, batches could be run daily or every few days.

5.9 Results Analysis and Summary

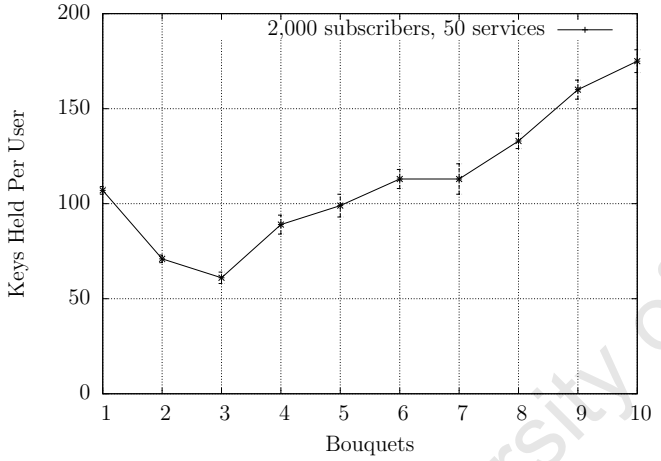
The simulation makes use of a simplified and somewhat artificial model of customer behaviour, with randomness introduced by a pseudo-random number generator. The objective of the simulation was to ascertain the basic viability and scalability of the proposed design for practical implementations. For this purpose visual analysis of



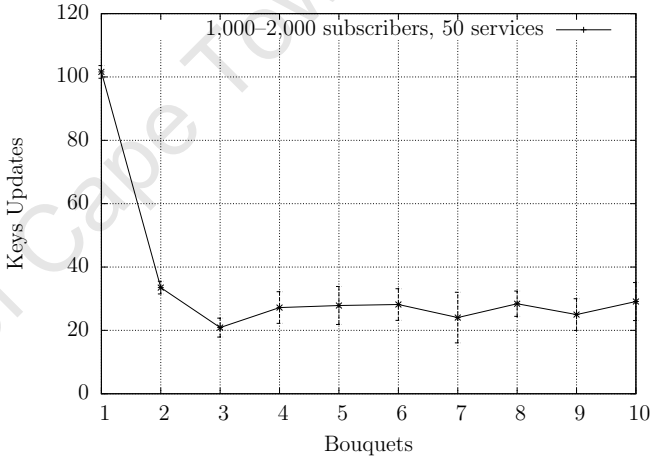
(a) Keys Held as Services Vary



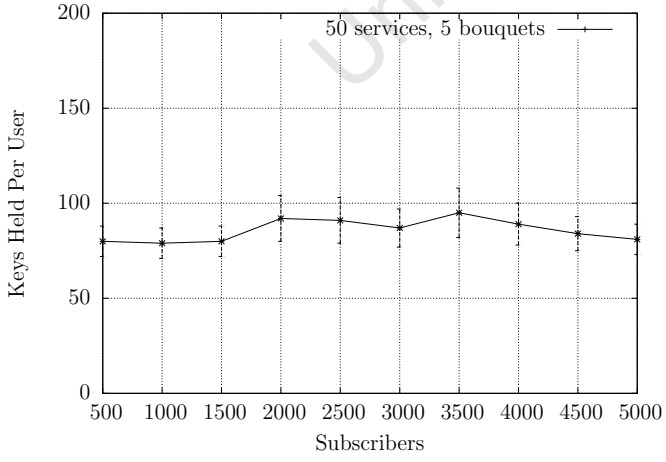
(b) Updates Processed as Services Vary



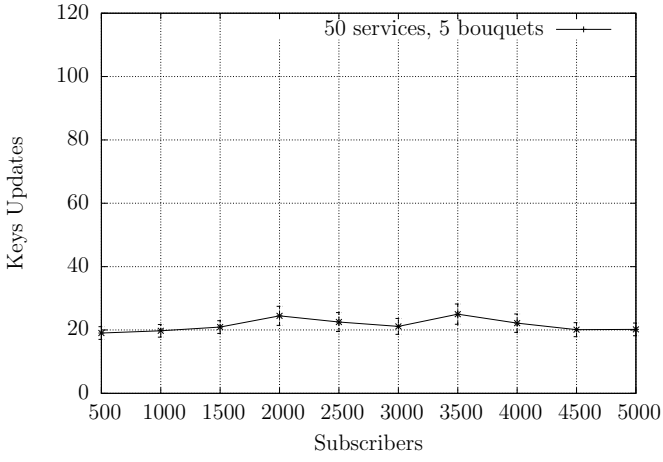
(c) Keys Held as Bouquets Vary



(d) Updates Processed as Bouquets Vary



(e) Keys Held as Subscribers Vary



(f) Updates Processed as Subscribers Vary

Figure 5.8: Average Keys Held and Updates Processed by a Client

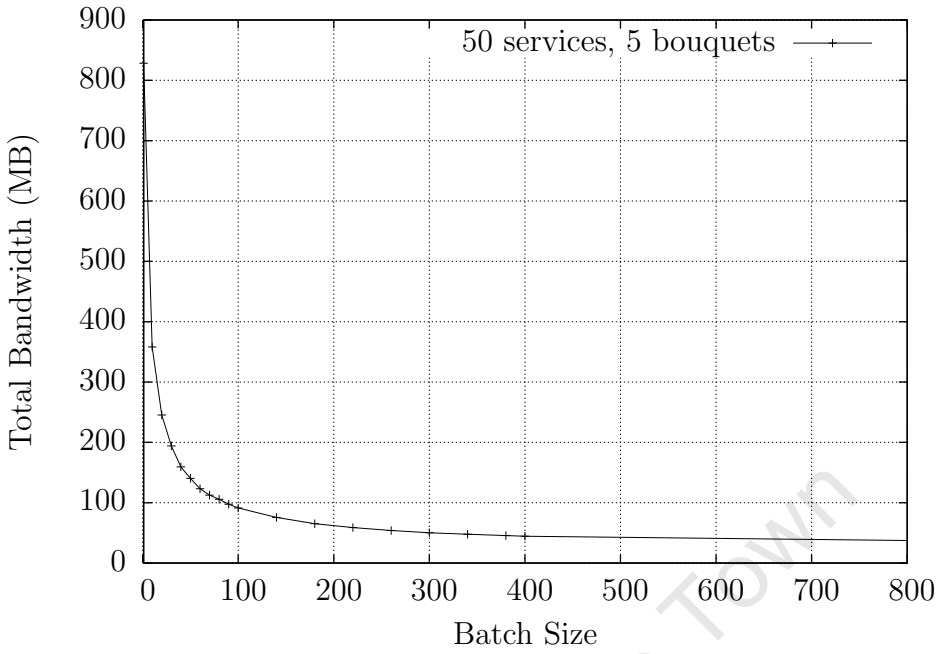


Figure 5.9: Effect of Batches on Outgoing Server Bandwidth. Data transfer required to increase subscribers from 50,000 to 100,000, with 10% turnover.

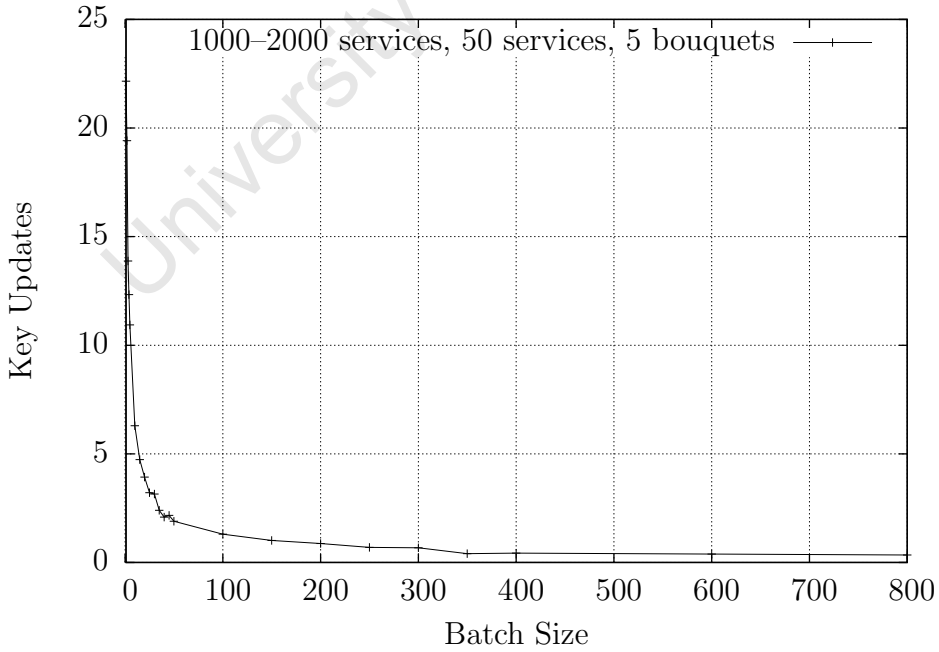


Figure 5.10: Effect of Batches on Average Number of Keys Processed by Clients per Add/Remove operation.

the graphed results was deemed sufficient and so detailed statistical analysis of the results was not performed.

Some undesirable inflections are visible in the graphed data, particularly in the client operations graphs. These are due to the limited number of simulations runs that were used to produce the data. The running time for simulating the computational operations of clients is significant and it was impractical to perform many runs for each datapoint. However for the stated objectives, the collected data is sufficient to demonstrate the operation of prototype system.

In Section 5.3, the memory consumption of the key server was assessed. It was observed to scale linearly with an increase in number of subscribers, as expected. The results of varying the degree of the LKH tree were unexpected. They demonstrated the need for careful memory management in a real implementation to avoid non-optimal memory alignment causing wasting space.

In Section 5.4, the overhead in memory consumption required to support the proposed multiple-service architecture was shown to be more efficient than the naïve approach for fewer than 15 bouquets. Also in this section, we showed that the grouping of subscribers into groups sharing different group keys incurs minimal overhead compared to all subscribers sharing a single group key.

Two alternative methods of expanding a full LKH tree were compared in Section 5.5. Our proposed "fast-grow" operation was shown to have a marginal improvement over the standard approach.

In Section 5.6 the average size of key updates was shown to be significantly improved by the use of LKH+ over LKH. This will be true of an environment with more subscribers joining than leaving, on average. The cost of client processing and storage was considered in Section 5.7.

Finally, batched key updates were shown in Section 5.8 to provide a significant reduction in number of key updates and associated bandwidth, even for small (50 to 100) batch sizes. A practical implementation should definitely consider the use of batching updates.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

The distribution network of a pay-television operator is protected cryptographically by means of a conditional access system. Such a system is responsible for sharing keys with authorised subscribers and provides a mechanism for updating those keys on a regular and timely basis to allow the subscribers to view content. The system needs to support large numbers of users and services for the pay-TV application. The key contribution of this thesis is to address the design of an end-to-end framework for a conditional access system in an IPTV environment.

Group key management techniques are applied to this problem. A prototype implementation produced experimental results that demonstrate it is possible to successfully develop a workable framework, which is usable as the basis for a commercial CA system. The implementation is highly scalable with number of subscribers and number of services. This is achieved by dividing services into a small number of groups that are treated as atomic units of subscription. This coincides with bouquets, an existing convenience mechanism in digital television for grouping services. Subscribers are grouped according to their selection of bouquets. While this does place restrictions on the flexibility of customers' choice, it is in-line with pay-TV operators' and their content providers' business practices — namely, bundling services to subsidise those that are of interest to a smaller audience with those of interest to a larger audience. This method improves upon an existing technique which supports completely flexible service selection but is only practical for a small number of services.

Additionally, the solution enforces subscription selections cryptographically. Subscribers can only decrypt those services for which they are authorised, as opposed to using a single key shared amongst all subscribers for all services — an approach which would only exclude non-subscribers and would depend on receiver software enforcing subscription selections.

An efficient and novel technique for growing full key trees was proposed that improves upon the performance of the standard technique.

The protocols that have been designed and presented do not require a smart card as a secure storage mechanism. A protocol for establishing a secret key shared between the server and an untrusted client was presented. All that is required is a trusted system that can verify the integrity of authenticated code and assure that is guaranteed not to disclose keys. The system is flexible enough to support video-on-demand content encrypted per-user and an extension supporting multiple devices per subscription was proposed.

6.2 Summary of Contribution

This thesis has presented a detailed consideration of the end-to-end requirements for designing a practical IPTV Conditional Access system. The specific, unique contributions of this research are as follows:

- A proposed collusion attack on the F-PPC key refreshment scheme [41]. This attack was presented in Section 3.3.6.
- A modification to the F-PPC multiple service architecture was presented in Section 4.4. This modification addresses the problems of scalability with increase in number of services that are present in that scheme (identified in Section 3.4.2).
- A fast key tree "growth" operation to increase the size of a full LKH+ [32] key tree that reduces the total number of key update messages and the number of messages that a receiver has to process over other techniques. This operation was described in Section 4.3.2.
- The design of a protocol for initial establishment of a key shared only between the key server and a receiver device, given in Section 4.5.

- The combination of LKH+ group rekeying with the aforementioned multiple-service architecture to support multiple services, each with independent encryption keys, divided into bouquets to allow subscriber choice. The key server operations, format of the multicast rekey message, and details of client processing of key updates were given in Sections 4.6, 4.7 and 4.8, respectively.
- "Application vaults", for securing persistent key storage on receiver devices, based on the "VAULTS" cryptographic access control scheme [31], was described in Section 4.12.
- The implementation of a prototype CA system and simulation framework to assess the viability of a practical implementation of our proposed system. The scalability of the system was assessed with respect to varying numbers of subscribers, services, and bouquets. Additionally, the effect of batched key updates was considered. The implementation details were given in Section 5.2 and analysis of the experimental results in Chapter 5.

6.3 Directions for Future Work

6.3.1 Software

Additional work on the software would require developing more comprehensive networking code, integration with a proper subscriber management system that possibly incorporates direct sign-up, integrating the server and client with a media streaming framework such as the Live555 libraries [25] (which support RTP, RTCP, and RTSP), modification to comply with the DVB-IPTV standard, and general performance and robustness improvements.

6.3.2 Green Technology

The current global trend is towards energy saving and "green" technology. From both a legislated and ethical standpoint, developing devices such as set-top-boxes that have minimal power consumption is highly desirable. A critical component of this is the *standby* power consumption of a device. That is, how much power it uses when in *standby* mode.

In the European Union, for example, legislation states that electrical devices sold after December 2012 should use no more than 0.5 Watts in standby mode (or 1 Watt if they display information such as a clock when in standby) [13].

This places restrictions on the operation of a set-top-box. Typically, satellite set-top-boxes in standby only disable their video and audio outputs and otherwise maintain full connections to the satellite network. This is in order to receive software updates, altered programme schedules, and most importantly, group key updates and other entitlement information. If such a set-top-box is properly powered-off for any period of time (often only possible by physically unplugging it), once turned on again it often takes quite a while to start descrambling video. This is because of the *carousel* fallback mechanism that is in place for recovering after missed key updates. In any system with no return-path, the only way to distribute potentially missed keys to clients is to continuously broadcast them.

The design of the conditional access system therefore has direct implications on the power usage and environmental impact of a set-top-box. While these power savings for a single set-top-box might seem insignificant, cumulatively they are very significant. A multiple-tuner, high-definition satellite PVR might use as a much as 100 Watts when operating. If this is reduced to 1 Watt when in standby, a savings of 99 Watts is achieved. Assuming the majority of set-top-boxes are in standby for 90% of their lifetime[1], in a network with 5 million subscribers a total power reduction of 495 MW is possible – almost a third of the total power generation capacity of the Koeberg Nuclear Power Station.

Maintaining a STB network connection and other processing functions with less than 1 Watt of power is not feasible with current technology. It should therefore be clear that any stateful conditional access system needs to rely on a fallback mechanism to supply missed key updates to devices that have been off or in standby. The time complexity of the aforementioned carousel mechanism is $O(n)$, for a system with n subscribers. In practice this creates a bad user experience as the waiting period for a system with a realistic subscriber base can be up to an hour – a very poor user experience if this occurs every time the STB is turned on. Overcoming this might require a reduction in the frequency of key updates, potentially resulting in a reduction of the security of the system.

In an IPTV network, a return path is available. A reliable fallback method is therefore for a client to establish a unicast connection to the key server and request

[1]This assumes that households watch 2 hours of television per day, on average.

the latest group key. However, dependant on the frequency of key updates, this will degrade to the situation where all clients need to establish a fallback connection every time they are powered on. This will negatively impact network bandwidth and place extremely high demand on the server.

A stateless conditional access system would avoid many of these problems. Additions or modifications to a technique such as SDR that accounts for the poor performance in the case of removed users would be a good direction for future work. Academic research in this area is on-going [48].

6.3.3 Network Simulation

More comprehensive network simulation with a system such as NS-3 would be relevant if detailed information about a real topology was available. This would need to include the operator network, ISPs and other intermediate service providers, end users' networks, and so forth. Such a simulation would allow measurements of traffic flow, and more accurate simulation of failures

6.4 Final Remarks

This research has successfully highlighted some primary problem areas in conditional access and has proposed workable solutions to address these problems. There is much scope for future work in extending the solution, and in fact in the greater problem of end-to-end content security.

Glossary

AES Advanced Encryption Standard.

CA Conditional Access.

CAS Conditional Access System.

CW Control Word.

DRM Digital Rights Management.

DTT Digital Terrestrial Television.

DVB Digital Video Broadcasting.

DVR Digital Video Recorder.

ECM Entitlement Control Message.

EMM Entitlement Management Message.

FTA Free-To-Air.

FTV Free-To-View.

IGMP Internet Group Management Protocol.

IPTV Internet Protocol Television.

IPv4 Internet Protocol version 4.

IPv6 Internet Protocol version 6.

LAN Local Area Network.

MLD Multicast Listener Discovery.

PKI Public-Key Infrastructure.

PPV Pay-Per-View.

PVOD Push-Video-On-Demand.

RTCP Real-time Transport Control Protocol.

RTP Real-time Transport Protocol.

RTSP Real Time Streaming Protocol.

SABC South African Broadcasting Corporation.

SKDC simple key distribution centre.

STB Set-Top-Box.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

VOD Video-On-Demand.

References

[1] Federal Information Processing Standards Publication (FIPS 46-3). Data Encryption Standard (DES), 1999.

[2] Federal Information Processing Standards Publication (FIPS 197). Advanced Encryption Standard (AES), 2001.

[3] Aciicmez, O., Seifert, J.-P., and Zhang, X. A Secure DVB Set-Top Box via Trusting Computing Technologies. In *6th IEEE Consumer Communications and Networking Conference, CCNC 2009* (Las Vegas, USA, January 2009), pp. 1–8.

[4] Azad, M., Ahmed, A., and Alam, A. Digital Rights Management. *International Journal of Computer Science and Network Security 10*, Number 11 (2010), pp. 24–33.

[5] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and Norrman, K. The Secure Real-time Transport Protocol (SRTP). *Request for comments 3711, Internet Engineering Task Force* (March 2004).

[6] Briscoe, B. MARKS: Zero side effect multicast key management using arbitrarily revealed key sequences. In *Proc. First International Workshop On Networked Group Communication, NGC'99* (Pisa, Italy, 1999), pp. 301–320.

[7] Cruickshank, H., Howarth, M., and Iyengar, S. A Comparison between satellite DVB conditional access and secure IP multicast. In *14th IST Mobile and Wireless Communications Summit* (Dresden, Germany, 2005).

[8] Cutts, D. J. DVB Conditional Access. *Electronics & Communication Engineering Journal 9*, Number 1 (1997), pp. 21–27.

[9] Deering, S. Host Extensions for IP Multicasting. *Request for comments 1112, Internet Engineering Task Force* (August 1989).

[10] Deering, S., Fenner, W., and Haberman, B. Multicast listener discovery (mld) for ipv6. *Request for comments 2710, Internet Engineering Task Force* (October 1999).

[11] Deering, S., and Hinden, R. Internet protocol, version 6 (ipv6) Internet Protocol, Version 6 (IPv6) Specification. *Request for comments 1883, Internet Engineering Task Force* (December 1995).

[12] Diffie, W., and Hellman, M. New directions in cryptography. *IEEE Transactions on Information Theory IT-22*, Number 6 (November 1976), pp. 644–654.

[13] European Commission. Commission regulation (ec) no 107/2009. *Official Journal of the European Union* (February 2009).

[14] European Telecommunications Standards Institute. Digital Video Broadcasting: Framing structure, channel coding and modulation for 11/12 GHz satellite services. European Standard EN 300 421, August 1997.

[15] European Telecommunications Standards Institute. Digital Video Broadcasting: Head-end implementation of DVB SimulCrypt. ETSI TS 103 197, October 2008.

[16] European Telecommunications Standards Institute. Digital Video Broadcasting: Transport of MPEG-2 TS Based DVB Services over IP Based Networks. ETSI TS 102 034, August 2009.

[17] European Telecommunications Standards Institute. Digital Video Broadcasting: Support for use of the DVB Scrambling Algorithm version 3 within digital broadcasting systems. ETSI TS 100 289, September 2011.

[18] Floyd, S., Jacobson, V., Liu, C., McCanne, S., and Zhang, L. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking 5*, Number 6 (December 1997), pp. 784–803.

[19] Haroon, J. Decentralized key management for large dynamic multicast groups using distributed balanced trees. Master's thesis, National University of Computer and Emerging Sciences, Lahore, Pakistan, 2004.

[20] Hellman, M. An Overview of Public Key Cryptography. *IEEE Communications Magazine 16*, Number 6 (November 1978), pp. 24–32.

[21] International Telecommunication Union. Security Architecture for Open Systems Interconnection for CCITT Applications. Recommendation X.800, 1991.

[22] International Telecommunication Union. Information Technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks. Recommendation X.509, 2008.

[23] Kerckhoffs, A. La cryptographie militaire. *Journal des sciences militaires IX*, pp. 5–38 (January 1883).

[24] Li, W., and Gu, D. Security Analysis of DVB Common Scrambling Algorithm. In *The First International Symposium on Data, Privacy, and E-Commerce* (November 2007), pp. 271–273.

[25] Live Networks, Inc. Live555 streaming media. `http://www.live555.com/liveMedia/` (last accessed 15 September 2013).

[26] McGrew, D. A., and Sherman, A. T. Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering 29*, Number 5 (May 2003), pp. 444–458.

[27] Montpetit, M., Mirlacher, T., and Ketcham, M. IPTV: An end to end perspective (invited paper). *Journal of Communications 5*, Number 5 (2010), pp. 358–373.

[28] Muller, R. DStv cracked and pirated by techies; numerous arrests, MyBroadband Tech News. `http://mybroadband.co.za/news/broadcasting/34606-dstv-cracked-and-pirated-by-techies-numerous-arrests.html` (last accessed 15 September 2013), September 2011.

[29] Naor, D., Naor, M., and Lotspiech, J. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology, CRYPTO 2001* (February 2001), Springer-Verlag, pp. 41–62.

[30] OpenSSL Project, The. OpenSSL: The Open Source toolkit for SSL/TLS. `http://www.openssl.org/` (last accessed 15 September 2013).

[31] Payne, C. A cryptographic access control architecture secure against privileged attackers. In *Proceedings of the 2007 ACM workshop on Computer security architecture, CSAW '07* (New York, USA, 2007), ACM, pp. 70–76.

[32] Perlman, R. "LKH+": Simplification of LKH, an observation from the conference floor.

[33] Pietro, R. D., Mancini, L. V., and Jajodia, S. Providing secrecy in key management protocols for large wireless sensors networks. *Ad Hoc Networks 1*, 4 (2003), pp. 455–468.

[34] Quinn, B., and Almeroth, K. IP Multicast Applications: Challenges and Solutions. *Request for comments 3170, Internet Engineering Task Force* (September 2001).

[35] Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, Number 2 (Feb. 1978), pp. 120–126.

[36] RSA Laboratories. *RSA Laboratories' Frequently Asked Questions about Today's Cryptography, Version 4.1.* RSA Security Inc., 2000.

[37] Setia, S., Koussih, S., Jajodia, S., and Harder, E. Kronos: A scalable group rekeying approach for secure multicast. In *IEEE Symposium on Security and Privacy* (Oakland, USA, May 2000).

[38] Setia, S., Zhu, S., and Jajodia, S. A comparative performance analysis of reliable group rekey transport protocols for secure multicast. In *Performance Evaluation, special issue on the Proceedings of Performance 2002* (2002), pp. 21–41.

[39] Speakman, T., Crowcroft, J., Gemmell, J., Farinacci, D., Lin, S., Leshchiner, D., Luby, M., Montgomery, T., Rizzo, L., Tweedly, A., Bhaskar, N., Edmonstone, R., Sumanasekera, R., and Vicisano, L. PGM Reliable Transport Protocol Specification. *Request for comments 3208, Internet Engineering Task Force* (2001).

[40] Stallings, W. *Cryptography and Network Security: Principles and Practice*, 5 ed. Prentice Hall, 2010.

[41] Sun, H., Chen, C., and Shieh, C. Flexible-Pay-Per-Channel: A New Model for Content Access Control in Pay-TV Broadcasting Systems. *IEEE Transactions on Multimedia 10*, Number 6 (October 2008), pp. 1109–1120.

[42] Vermeulen, J. Subscriber selected DStv channels demanded, MyBroadband Tech News. `http://mybroadband.co.za/news/broadcasting/`

56271-subscriber-selected-dstv-channels-demanded.html (last accessed 15 September 2013), July 2012.

[43] Wallner, D. M., Harder, E. J., and Agee, R. C. Key management for multicast: Issues and architectures. *Request for comments 2627, Internet Engineering Task Force* (June 1999).

[44] Weinmann, R.-P., and Wirt, K. Analysis of the DVB Common Scrambling Algorithm. In *Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, CMS 2004* (2004), Kluwer Academic Publishers, pp. 195–207.

[45] Wong, C. K. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking 8*, Number 1 (2000), pp. 16–30.

[46] Yang, Y., Li, X., Zhang, X., and Lam, S. Reliable group rekeying: design and performance analysis. In *ACM SIGCOMM 2001* (San Diego, USA, August 2001), pp. 27–38.

[47] Zetter, K. From the eye of a legal storm, Murdoch's satellite-TV hacker tells all, Wired. http://www.wired.com/politics/security/news/2008/05/tarnovsky (last accessed 15 September 2013), 2008 May.

[48] Zhu, S., and Jajodia, S. Scalable group key management for secure multicast: A taxonomy and new directions. In *Network Security*, S. C. H. Huang, D. MacCallum, and D. Du, Eds. Springer US, 2010, pp. 57–75.

[49] Zhu, S., Setia, S., and Jajodia, S. Adding reliable and self-healing key distribution to the subset difference group rekeying method. In *Group Communications and Charges: Technology and Business Models. Proceedings of the 5th COST 264 International Workshop on Networked Group Communications, NGC 2003* (2003), Springer-Verlag, pp. 107–118.